VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vývoj pro platformu iOS
Development for iOS Platform

2015 Martin Púčik

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Púčik**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Vývoj pro platformu iOS
Development for iOS Platform

Zásady pro vypracování:

Platforma iOS patří mezi nejvýznamnější hráče na poli mobilních systémů. Cílem této práce je ilustrovat možnosti této platformy, a to především z pohledu vývojového prostředí, procesům, apod.

1. Popište vývojové prostředí platformy iOS, a to především z pohledu vývojových nástrojů, procesů a požadavků.
2. Navrhněte aplikaci tak, aby ilustrovala klíčové prvky platformy.
3. Implementujte mobilní aplikaci a popište detailně samotný vývojový proces. Tento proces zdokumentujte např. postupnými verzemi projektu, videonávody, atd.
4. Zahrňte do popisů a dokumentace také fáze testování, distribuce a uživatelské zpětné vazby.
5. Zhodnoťte platformu iOS jako celek z pohledu vývojáře.

Seznam doporučené odborné literatury:

[1] P. Buttfield-Addison: Learning Cocoa with Objective-C: Developing for the Mac and iOS App Stores, 2012, O'Reilly Media, ISBN: 978-1449318499
[2] J. Conway: iOS Programming: The Big Nerd Ranch Guide, 2013, Big Nerd Ranch Guides, ISBN: 978-0321942050

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Radecký, Ph.D.**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostravě 6. 5. 2015 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 6. 5. 2015 .....

Rád by som sa poďakoval Ing. Michalovi Radeckému, Ph.D. za jeho pomoc a odborné rady pri vypracovaní mojej bakalárskej práce.

Abstrakt

Táto práca je rozdelená do 4 častí. V prvej kapitole popísané vývojové prostredie pre platformu iOS. Druhá kapitola sa zaoberá podrobným popisom priebehu návrhu samotnej aplikácie. Tretia kapitola vysvetľuje implementáciu.

Kľúčové slová: Xcode, Objective-C, iOS, Apple, League of Legends

Abstract

This thesis is devided into four chapters. In the first chapter is described whole developing enviroment for Apple iOS platform. Second chapter deals with detailed description of designing the aplication. In third chapter takes place detailed implementation of key functions. Fourth chaper shows testing phases of aplication and in the last section is the Apple iOS platform summarized mainly from developer point of view.

Keywords: Xcode, Objective-C, iOS, Apple, League of Legends

Zoznam použitých skratiek a symbolov

OS	–	Operačný systém
multitask	–	proceses are performed during the same period of time
IDE	–	Integrované vývojové prostrediet
PC	–	Osobný počítač
VPS	–	Virtual Private Server
SDK	–	Software development kit
LoL	–	League of Legends

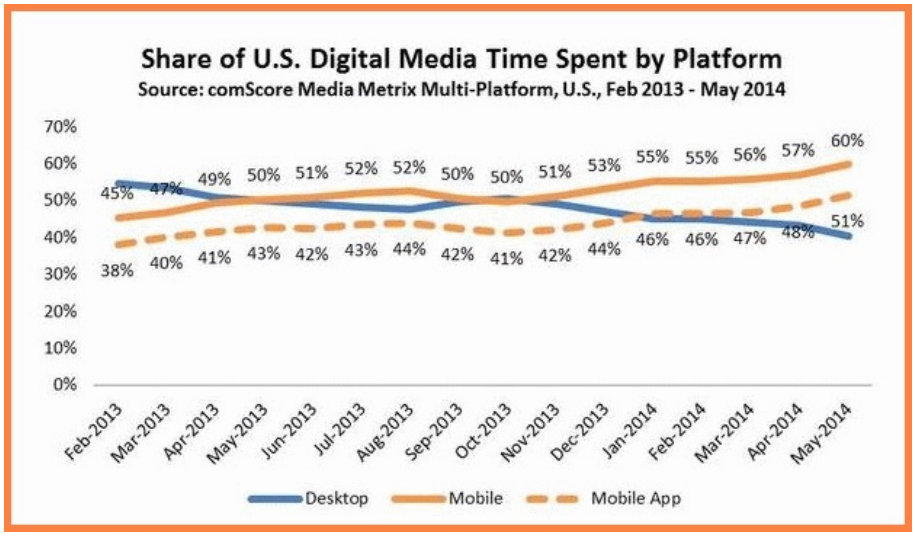
Obsah

Seznam obrázků

Zoznam výpisov zdrojového kódu

1 Úvod

V roku 2014 boli mobilné telefóny (*smartphones*) a tablety zodpovedné za viac ako 60% webovej komunikácie [1]. Viac ako 51% z tejto komunikácie bolo z mobilných aplikácií s multimediálnym obsahom ako napríklad *audiostreaming* služby *Pandora* alebo *Spotify*, komunikačné aplikácie ako *WhatsApp* alebo *Viber* alebo aplikácie sociálnych sietí *Facebook* alebo *Twitter*.



Obrázek 1: Webová komunikácia v rokoch 2013/14 z pohľadu platformy

Pre tento vývoj mobilného trhu použila firma Apple termín *Post-PC Era*, ktorý prvýkrát použil David D. Clark v 1999, ktorým popisuje pokles predajov PC a presun zákazníkov k mobilným zariadeniam a tabletom, ktoré sú vďaka *cloud* synchronizácii zbavené závislosti na PC a pomocou aplikácií dokážu nahradiť aj PC funkcionalitu.

Za rok 2014 mali dva najrozšírenejšie mobilné operačné systémy *Android* a *iOS* spoločný podiel na trhu 96,3% [2], čím sa stali najbežnejšou voľbou koncových zákazníkov. Android dosiahol 81.5% trhový podiel nielen vďaka viac ako miliarde predaných zariadení, ale aj vďaka svojej open-source lincencii. iOS získal 14.8% z celkového trhu pri približne 192 miliónoch predaných zariadení firmy Apple.

Za rovnaké obdobie obchod s aplikáciami *Google Play* operačného systému Android vyprodukoval o 60% viac stiahnutí ako zaznamenal *App Store* operačného systému iOS ale aj napriek tomu vygeneroval o 70% viac príjmov ako Android a zostal tak lukratívnejšou platformou pre vývojárov.[3]

1.1 Čo je iOS

iOS je mobilný operačný systém firmy Apple určený pre ich mobilné zariadenia. Aktuálna verzia iOS je 8, pričom Apple udržiava jeho životný cyklus každoročnými aktualizáciami.

Tieto aktualizácie sú dostupné v rovnakom čase pre všetky aktuálne ponúkané zariadenia firmy Apple a sú distribuované OTA (*Over the Air*) alebo pomocou funkcie Aktualizovať (*Update*) v aplikácii *iTunes*. Táto aplikácia je predinštalovaná na zariadeniach s operačným systémom Mac OS X a je voľne dostupná pre platformu Windows. Každá nová verzia alebo aktualizácia je zdarma pre všetky podporované zariadenia. Nové funkcie sú dostupné pre všetky zariadenia bez obmedzení s výnimkou funkcií, ktoré vyžadujú špecifický hardware.

1.2 Apple Developer Program

Pre vývoj na Apple platformy iOS a Mac OS X je potrebné mať aktívny a zakúpený *Apple Developer Program*[4]. Tieto programy ponúkajú kompletnú sadu technických prostriedkov, podporu a prístup k beta verziám softvéru a operačných systémov.

Apple Developer Program je rozdelený na 2 časti:

- iOS Developer Program
- Mac Developer Program

Každá časť je spoplatnená osobitne cenou $99 USD a dĺžka platnosti programu je 1 rok. Na registráciu Apple Developer Programu je potrebné mať existujúci Apple ID účet a platnú kreditnú kartu. Pre celkový vývojový proces Apple Developer Program nie je potrebný, vývojové prostredie *Xcode* je možné používať aj bez aktívneho programu, určité funkcie a služby ho však vyžadujú.

- **Testovanie na iOS zariadeniach** [5]
  Pre spustenie vytvorenej aplikácie na reálnom iOS zariadení je potrebný aktívny program a dané zariadenie musí byť zaregistrované na Apple Developer portáli a priradené k podpisovaciemu certifikátu. Bez aktívneho programu je možné testovanie aplikácie len v simulátore integrovanom vo vývojovom prostredí Xcode.
- **Distribúcia aplikácie do AppStore** [6]
  Aktívny Developer Program je nutný na distrubúciu aplikácie do obchodu App Store.
- **Prístup do iTunes Connect portálu** [7]
  iTunes Connect je portál na kompletnú správu aplikácií, beta testovania, In-App nákupov, finančných kontraktov a zmlúv, vývojárskych účtov a hodnotení aplikácií.

2 Xcode - vývojové prostredie iOS

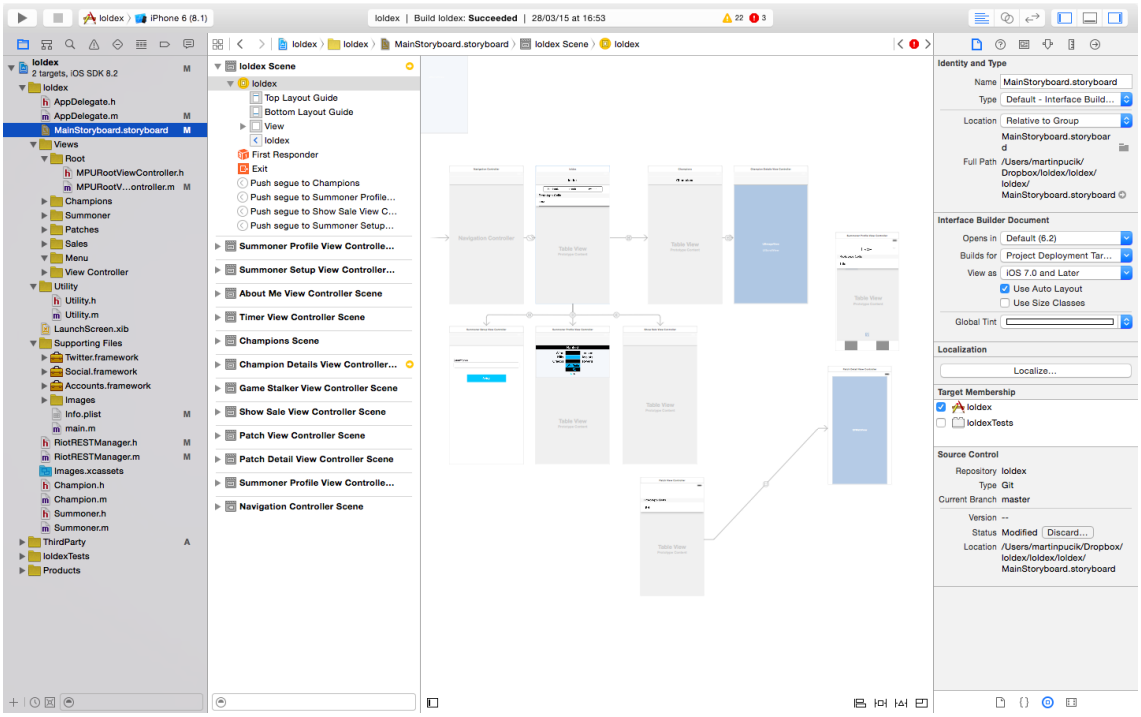
Xcode je integrované vývojové prostredie vytvorené firmou Apple, ktoré je používané ako primárny nástroj na vytvorenie aplikácií pre Apple produkty ako iPhone, iPad alebo Mac. Xcode obsahuje nástroje potrebné na kompletný vývojový proces - od vytvorenia aplikácie, cez jej testovanie, optimalizáciu a odoslanie na schválanie do App Store.

Xcode integruje úpravu kódu, dizajnu, správu zdrojov, testovanie a ladenie v jednom okne pracovného prostredia. Celý Xcode workspace sa dynamicky prispôsobuje aktuálnej úlohe, na ktorej programátor pracuje. Toto prostredie je vysoko prispôsobiteľné potrebám programátora.

Xcode je dostupný zdarma a je distribuovaný cez integrovaný obchod App Store na každom zariadení Mac alebo cez vývojársky portál Apple Developer [1].

2.1 Rozhranie

Rozhranie Xcode (Workspace window) slúži ako primárny nástroj na vytvorenie a správu projektov, ako základných jednotiek vývoja. Projekt obsahuje a udržiava vzťahy medzi všetkymi elementami potrebnými na vytvorenie aplikácie, frameworku, plug-inu alebo akéhokoľvek software produktu.



Obrázek 2: Xcode rozhranie

[1] Adresa portálu: http://developer.apple.com

2.1.1 Editor

Workspace window vždy obsahuje Editor(Editor area). Po výbere akéhokoľvek súboru z projektu sa obsah daného súboru zobrazí v príslušnom type editoru.

- **Source editor**
  Plnohodnotný editor zdrojového kódu. Podporuje jazyky C, C++, Obj-C a Swift.
- **Interface Builder**
  Grafický editor určený na tvorbu užívateľského rozhrania
- **Project editor**
  Editor, ktorý zobrazuje konfiguračné XML súbory projektu s koncovkou .plist v užívateľsky prívetivejšom rozhraní. V týchto súboroch sa definujú špecifické pravidlá, ako má byť aplikácia zostavená, ako napríklad možnosti build procesu, typy podporovaných architektúr, na ktoré je aplikácia skompilovaná alebo aj názvy aplikácie, ktoré sa zobrazia na rôznych miestach ako domáca obrazovka zariadenia alebo názov v App Store.

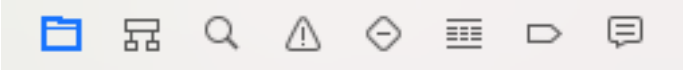
Editor je možné prispôsobiť pomocou nastavovacích tlačidiel na pravej strane panela nástrojov:

Obrázek 3: Nastavenia editora

- **Standard editor**
  Vyplní celý editor obsahom vybraného súboru.
- **Assistant editor**
  Rozdelí prostredie editoru na 2 vodorovne susediace editory s logicky naväzujúcim obsahom. Obsah každej časti rozdeleného editoru sa dá jednotlivo nastaviť.
- **Version editor**
  Rovnako ako Assistant editor rozdelí prostredie editoru na 2 editory. Version editor zobrazí zmeny medzi vybraným súborom v ľavom editore a inou verziou vybraného súboru v pravom editore. Tento typ editoru funguje len v prípade, že má projekt zapnutý *Source Control*

2.1.2 Navigator area

Bočný panel Navigátora sa používa na prístup k súborom, symbolom, testovacím jednotkám, diagnostike a ostatným súčastiam projektu. Pomocou panelu navigácie (*Navigation bar*) 4 je možné vybrať si z typu Navigátora, ktorý je najviac vhodný na aktuálne vykonávanú úlohu. Obsah panela sa prispôsobuje vybranému typu Navigátora.



Obrázek 4: Panel navigácie (*Navigation bar*)

- **Project navigator**
  Slúži na pridávanie, odstraňovanie, zoskupovanie a inú správu súborov v projekte a na výber súborov, ktoré majú byť v zobrazené alebo upravené v oblasti Editoru.

- **Symbol navigator**
  Zobrazí všetky symboly v projekte ako hierarchiu alebo zoznam. Možnosťou je filtrovanie v kombinácii len tried a protokolov, len symbolov v projekte alebo len kontainerov.

- **Find navigator**
  Pomocou funkcie hľadania a filtrov nájde a vypíše výskyt akéhokoľvek reťazca znakov v projekte.

- **Issue navigator**
  Vypíše všetky problémy nájdené počas diagnostiky, varovania a chyby nájdené počas otvárania, analýzy alebo kompilácie projektu.

- **Test navigator**
  Slúži na vytvorenie, správu, beh a následné posudzovanie testovacích jednotiek.

- **Debug navigator**
  Zobrazí všetky bežiace vlákna a priradené *stack* informácie v danom špecifikovanom čase počas vykonávania programu.

- **Breakpoint navigator**
  Rozšíri *breakpointy* špecifiovaním ich vlastností a charakteristík, ako napríklad nastavením podmienok spustenia daného *breakpointu*.

- **Report navigator**
  Zobrazí históriu kompilácií, behu, debugovania aplikácie, úlohy v správe verzií (*Source Control*) a kompletné štatistiky z *Continuous Integration*.

2.1.3 Utilities area

Oblasť nástrojov v pravom rohu pracovného okna Xcode ponúka rýchly prístup k prostriedkom Inšpektor (*Inspector*) a Knižnice (*Libraries*). Vyššie umiestnené okno v oblasti Nástrojov je Inšpektor a nižšie umiestnené okno ponúka prístup ku Knižnici.

Panel Inšpektora ponúka možnosť vybrať si najviac vhodný typ Inšpektora. Vždy sú zorazené minimálne dva typy:

- **File Inspector**
  Zobrazenie a úprava vlastností a metadát otvoreného súboru v *Editore*. Typicky sa používa na lokalizáciu storyboard a iných mediálnych súborov a zmenu nastavení pre súbory užívateľského rozhrania.

- **Quick Help**
  Zobrazí detaily o symbole, elemente rozhrania alebo o špecifickom nastavení kompilácie projektu. Zobrazí tiež stručný popis vybranej metódy v editore, kde a ako je táto metóda deklarovaná, jej rozsah, potrebné parametre, platformovú a architekturickú dostupnosť.

Knižnica poskytuje prístup k predpripraveným zdrojom na použitie v projekte:

- **File templates**
  Šablóny pre najbežnejšie typy súborov a štruktúry kódu.

- **Code snippets**
  Krátke kúsky zdrojového kódu pre použitie v projekte, ako napríklad deklarácie tried, deklarácie blokov alebo šablóny implementácií pre bežne používané Apple technológie.

- **Objects**
  Položky objektov používané pri vytváraní užívateľského rozhrania aplikácie.

- **Media**
  Všetky súbory, ktoré obsahujú grafiku, ikony, zvukové súbory a podobne.

Použitie knižnice je veľmi jednoduché pomocou *drag and drop* gesta na vybranú časť knižnice a presunutím do vhodnej časti pracovného okna Xcode. Napríklad pre použitie *snippet* kódu, stačí presunúť objekt reprezentujúci daný kód z knižnice do editora zdrojového kódu. Na vytvorenie nového zdrojového súboru je potrebné presunúť šablónu do časti navigátora projektu.

Knižnica obsahuje aj vyhľadávacie pole, ktoré filtruje zobrazené objekty vo vybranej časti podľa vloženého reťazca znakov. Napríklad slovo *button* zobrazí všetky typy tlačidiel, ktoré je možné použiť pri tvorbe užívateľského rozhrania.

2.1.4 Workspace toolbar

Panel na hornej strane pracovného okna Xcode poskytuje rýchly prístup k veľmi často používaným príkazom. "*Play*" tlačidlo skompiluje a spustí vybranú produkt z projektu. "*Stop*" tlačidlo ukončí aktuálny beh aplikácie. Menu schémy dovolí nakonfigurovať kompiláciu a beh vybraného produktu. Oblasť aktivity zobrazí pokrok vo vykonávaní aktuálnej úlohy pomocou stavovej správy, grafickej interpretácie pokroku, ikony varovaní a ikony chýb.

Obrázek 5: Panel pracovného okna Xcode (*Workspace Toolbar*)

2.2 Správa prostriedkov projektu

Aplikácie vytvorené v Xcode vyžadujú *projekt* ako základnú jednotku, ktorý obsahuje a udržiava organizované všetky potrebné súbory a zdroje. Nový projekt sa vytvorí pomocou *File » New » New Project* v Menu a v novom pracovnom okne sa zobrazí ponuka s výberom vstavaných šablón pre najbežnejšie typy aplikácií pre iOS a Mac OS X. Tieto šablóny obsahujú základné nastavenia projektu a súbory, ktoré pomáhajú začať vývojový proces rýchlo a efektívne.

2.2.1 Projekt ako repozitár zdrojov

Projekt obsahuje všetky elementy potrebné k tvorbe aplikácií alebo iných softvérových produktov ako nástroje príkazového riadku či plug-inov a udržiava vzťahy medzi týmito elementami, ktoré môžu obsahovať:

- Odkazy na súbory zdrojového kódu, implementačné aj hlavičkové súbory, knižnice, frameworky, obrázkové súbory a súbory užívateľského rozhrania.
- Skupiny a priečinky na organizáciu súborov v navigátore projektu.
- Nastavenia týkajúce sa kompilácie na projektovej úrovni.
- Ciele (*Targets*), kde každý cieľ produkuje práve jednu aplikáciu.

Výberom projektu v navigátore sa otvorí príslušný projektový editor, v ktorom je možné upraviť každý aspekt ako má byť aplikácia zostavená, od definovania verzie SDK po špecifikáciu možností kompilácie.

Po vytvorení nového projektu, Xcode automaticky vytvorí dve štandardné konfigurácie kompilácie projektu: **debug** a **release**. Tieto konfigurácie sa líšia v obsahu informácie z ladenia (*debug*) a stupňom optimalizácie a sú dostačujúce pre potreby vývoja a väčšina vývojárov necíti potrebu meniť akékoľvek z týchto nastavení.

2.2.2 Aplikovanie špecifických nastavení aplikácie

Každý projekt obsahuje minimálne jeden cieľ (*target*). Cieľ špecifikuje vytváraný produkt, ako napríklad či bude produktom iOS alebo Mac OS aplikácia. Pre úpravu nastavní cieľa, je potrebné otvoriť nastavenia projektu v editore, kde sú tieto nastavenia umiestnené pod položkou *Target*.

V okne *General* nastavení cieľa sú zobrazené nastavenia, ktoré sa často neupravujú a väčšina z nich bola nastavená na iných miestach počas vývojového procesu, ako napríklad vo vytváracích dialógoch pri zakladaní projektu.

Pre iOS aplikáciu okno *General* obsahuje nasledujúce nastavenia cieľa:

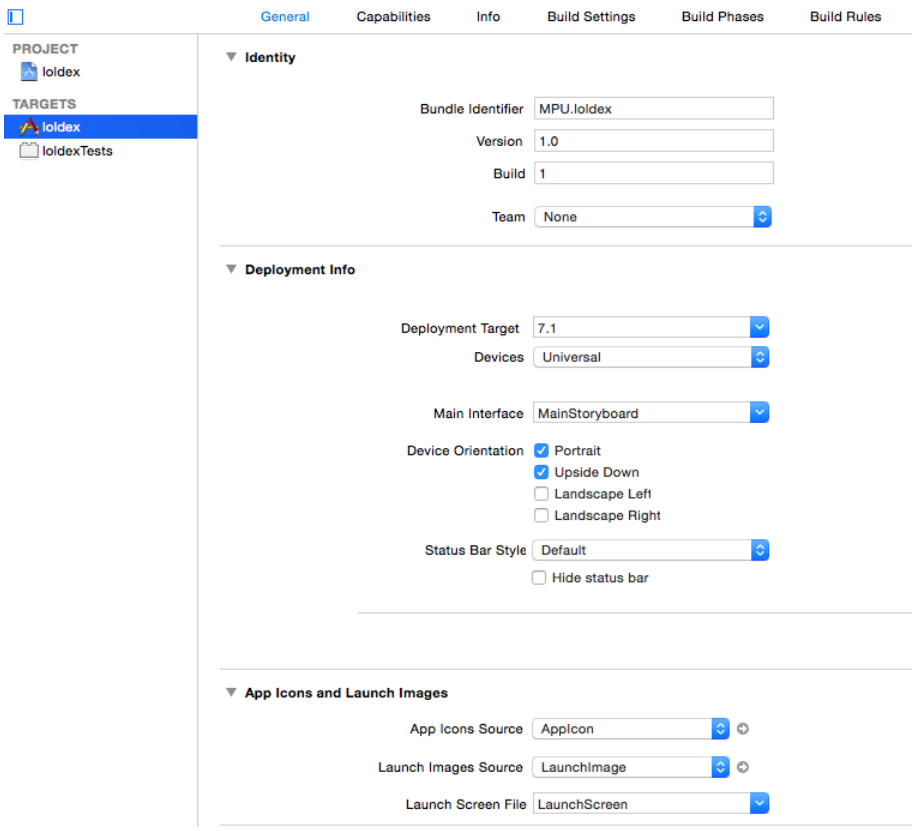
- **Bundle Identifier**
  Unikátny reťazec znakov, ktorý identifikuje aplikáciu pre operačný systém (iOS alebo Mac OS X) a pre App Store.
- **Version**
  Číslo verzie, pod ktorým sa aplikácia publikuje.
- **Build**
  Číslo, ktoré identifikuje konkrétny *build* aplikácie.
- **Team**
  Apple Developer Program tým, ktorého profil sa používa na kompiláciu a podpísanie produktu a následnú distribúciu aplikácie do App Store.
- **Deployment Target**
  Špecifikuje akú najstaršiu verziu OS aplikácia podporuje.
- **Devices**
  Špecifikuje typy zariadení, na akých aplikácia natívne pobeží. Možnosti sú *iPhone, iPad a Universal*.
- **Main Interface**
  Predvolený súbor užívateľského rozhrania, používaný pri spustení aplikácie.
- **Device Orientation**
  Orientácie užívateľského rozhrania, ktoré aplikácia podporuje. Možnosti sú *Portrait, Upside Down, Landscape Left a Landscape Right*.

2.2.3 Rozšírenie funkčnosti aplikácie

Na pridanie rôznych technológií Applu, ako napríklad *iCloud, Game Center, In-App, Maps* slúži sekcia *Capabilities*, kde sú zobrazené všetky dostupné technológie pripravené na pridanie do projektu. Na ich pridanie stačí prepnúť spínač pri vybranej technológii do polohy *ON* a do projektu sa pridajú všetky potrebné súbory a frameworky, ktoré daná technológia vyžaduje pre svoje fungovanie.

V niektorých prípadoch sa môže vyskytnuť problém pri pridávaní novej technológie. Táto chyba je potom popísaná pod danou technológiou v sekcii *Capabilities*.

Pred zapnutím je podrobne popísaná funkcionalita technológie a akcie, ktoré sa prevedú po jej zapnutí. U zapnutých technológií je možné upraviť nastavenia a vyriešiť prípadné vzniknuté chyby.



Obrázek 6: Nastavenia cieľa (*Target's General Settings*)

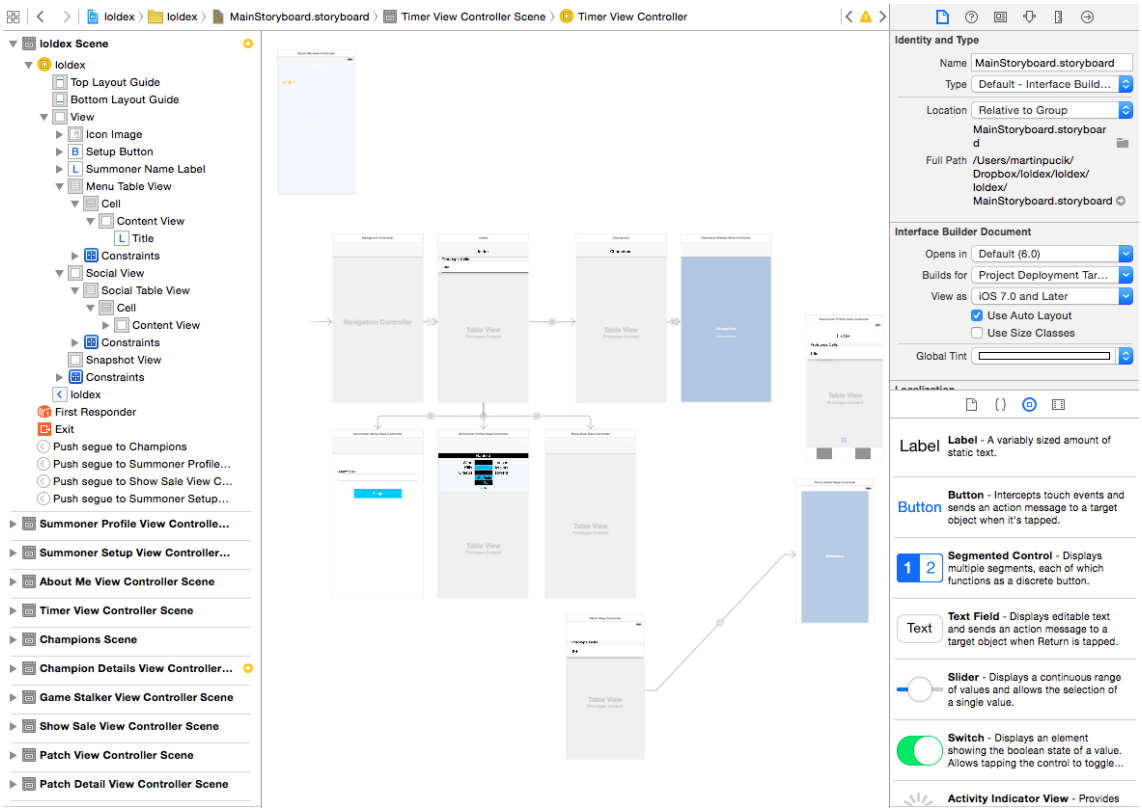
2.3 Tvorba užívateľského rozhrania aplikácie

Užívateľské rozhranie je vytvárané v editore typu *Interface Builder*. Po výbere súboru typu *.xib* alebo *.storyboard* z navigátora sa zdrojový kód preloží do jeho grafickej interpretácie a zobrazí v oblasti editora pracovného okna. Súbor typu *.xib* väčšinou predstavuje jednu obrazovku (*View Controller*). *Storyboard* špecifikuje niekoľko obrazoviek a prechodov (*segues*) medzi nimi. Na rozdiel od *.xib* súboru, môže *storyboard* obsahovať všetky grafické prvy a objekty užívateľského rozhrania. V predpripravených šablónach sa nachádza súbor užívateľského rozhrania, najbežnešie *.storyboard* súbor.

2.3.1 Pridanie UI elementov

Interface Builder je rozdelený na dve hlavné časti: *dock* na ľavej strane editora, ktorý obsahuje zoznam všetkých prvkov v súbore užívateľského rozhrania a vpravo *canvas*, kam sa UI objekty vkladajú.

Dock zobrazuje všetky použité objekty v stromovej štruktúre, ktorá interpretuje ktoré rodičovské objekty zapuzdrujú svoje detské objekty a ktorý objekt je najviac v popredí



Obrázek 7: *Interface Builder*

(najvyššie v *stacku*) vrámci svojho rodiča. Pre *.xib* súbory sa dá stromová štruktúra docku zmeniť na ikonovú reprezentáciu objektov pomocou tlačidla *Show/Hide Document Outline*.

Pre *.storyboard* súbory predstavuje najvyššie postavený rodičovský objekt v *docku* samotnú obrazovku (*View Controller*) alebo scénu (*scene*) zobrazenú na *canvase*.

Pridanie objektu do užívateľského rozhrania aplikácie prebieha z oblasti nástrojov (*Utilities area*) a z okna Knižnica objektov *Object library* . Pomocou jednoduchého *drag and drop* gesta je vybraný objekt umiestnený na *canvas* editoru. Pri vkladaní objektu sa zobrazia modré vodiace línie, ktoré pomáhajú správnemu umiestneniu do rodičovského objektu. Pre presnejšie zaradenie pridávaného objektu do hierarchie už existujúcich objektov sa dá toto gesto rovnako aplikovať aj v *dock* sekcii. Po pridaní alebo výbere objektu V *Interface Builder* editore, sa v okne inšpektora (*Inspector pane*) v oblasti nástrojov zobrazia možnosti nastavenia vzhľadu a vzťahov vybraného objektu.

2.3.2 Auto-Layout

Auto Layout je funkcia *Interface Builder* editora, ktorá umožňuje užívateľskému rozhraniu dynamicky sa prispôsobiť rôznym veľkostiam okien, displejov a orientáciám zariadení.

Pomocou definovania vzťahov (*constraints*) medzi objektami, *Auto Layout* nastaví polohu a veľkosť všetkých UI prvkov.

S *Auto Layoutom* všetky UI elementy zmenia svoju veľkosť a pozíciu keď:

- Užívateľ zmení orientáciu iOS zariadenia.
- Užívateľ zmení veľkosť okna Mac OS X aplikácie.
- Zmení sa veľkosť obsahu objektu, ako napríklad dĺžka textv tlačidle.

Pomocou *Control-drag* gesta (držanie klávesy Ctrl počas kliknutia na objekt) medzi dvoma objektami užívateľského rozhrania sa zobrazia možnosti dostupných vzťahov (*constraints*). Toto gesto je možné rozšíriť o klávesu Shift, ktorá dovolí vybrať niekoľko vzťahov naraz.

Zoznam všetkých vzťahov pre jednotlivé obrazovky (*View Controllers*) je zobrazený v sekcii *dock* v oblasti editoru. Jednotlivé vzťahy sú znázornené rovnako modrými vodiacimi čiarami v *canvas* časti editoru. Zoznam všetkých vzťahov pre vybraný objekt je zobrazený v okne inšpektora pod záložkou inšpektora veľkosti (*Size Inspector*).

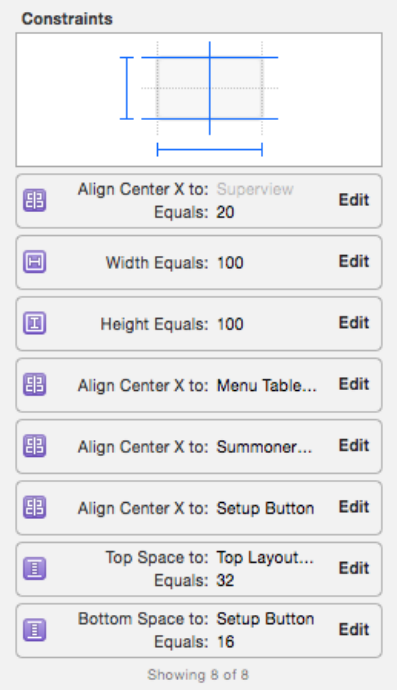
Horná časť tohto inšpektora graficky znázorňuje všetky typy *constraints* pre vybraný objekt a každá modrá čiara znázorňuje všetky vzťahy daného typu. Tieto typy môž byť:

- **Top Space** je umiestnenie horného okraja objektu.
- **Bottom Space** je umiestnenie dolného okraja objektu.
- **Leading Space** je umiestnenie ľavého okraja objektu.
- **Trailing Space** je umiestnenie pravého okraja objektu.
- **Center Horizontally In Container** je horizontálne centrovanie objektu.
- **Center Vertically In Container** je vertikálne centrovanie objektu.
- **Equal Widths** je šírka objektu vzhľadom na iný objekt.
- **Equal Heights** je výška objektu vzhľadom na iný objekt.
- **Aspect Ratio** je pomer vybranej veľkosti objektu vzhľadom na veľkosť iného objektu.

V dolnej sekcii *Size Inspectora* sa po výbere typu zobrazia všetky vzťahy a v tejto sekcii je možnosť ich nastaviť.

2.3.3 Prepojenie UI objektov so zdrojovým kódom

S objektami vytvorenými v editore *Interface Builder* je možné pracovať pomocou akcií (*actions*) a prepojení (*outlet connections*). Akcie sa používajú v prípade, že je potrebné aby určitý ovládací prvok z užívateľského rozhrania poslal správu do zdrojového kódu.



Obrázek 8: *Constraints*

Ovládací prvok (*control*) je UI objekt, ktorý okamžite pošle *action* správu alebo spôsobí viditeľnú zmenu po interakcii zo strany užívateľa s týmto objektom. Táto akcia vyvolá vykonanie danej časti zdrojového kódu. Bežne používané ovládacie prvky v iOS sú textové polia (*Text Fields*), posuvníky (*Sliders*), prepínače (*Switches*), tlačidlá (*Buttons*).

Outlety fungujú na presne opačnom princípe. Pomocou nich je možné poslať správu objektu v užívateľskom rozhraní. Prijímateľom tejto správy môže byť akýkoľvek objekt obsiahnutý v *.storyboard* alebo *.xib* súbore.

Najefektívnejším spôsobom, ako prepojiť objekty užívateľského rozhrania so zdrojovým kódom je zapnúť a použiť *Assistant editor*. V jeho ľavom okne je potrebné otvoriť *.storyboard* alebo *.xib* súbor a v jeho pravom okne hlavičkový alebo implementačný súbor vybranej triedy, ktorá slúži ako *controller* pre danú obrazovku. Následne pomocou *Control-drag* gesta (držanie klávesy Ctrl pri kliknutí na objekt) prepojíme vytvorený vzťah z *Interface Builder* okna do okna editoru a umiestnime prepojenie na miesto, kde chceme jeho deklaráciu. Po pustení zlačítka myši sa zorbazí menu, v ktorom nastavíme typ prepojenia (*action* alebo *outlet*) a jeho názov.

Po potvrdení prepojenia sa v prípade akcie vytvorí v implementačnom súbore kostra metódy s návratovým typom *IBAction*. Tento typ indikuje, že táto metóda môže byť prepojená s objektom v *.xib* alebo *.storyboard* súbore. Do vytvorenej kostry metódy je už len potrebné implementovať požadovanú logiku po interakcii s prepojeným objektom. O samotnú komunikáciu medzi objektami je postarané priamo systémom.

2.4 Spustenie aplikácie

Na zostavenie a spustenie iOS a Mac aplikácie je potrebné vybrať si schému a cieľ behu aplikácie v paneli nástrojov pracovného okna a kliknúť na tlačidlo *Run*. Tlačidlom *Stop* je možné beh aplikácie kedykoľvek ukončiť. V priípade iOS, spustenie aplikácie prebehne buď v simulátore priamo na Mac počítači alebo na iOS zariadení k Mac počítaču pripojenému. V prípade Mac aplikácie sa spustí priamo na Mac prístroji.

2.4.1 Výber build schémy

Schéma je zoskupenie nastavení, ktoré špecifikujú ktorý cieľ projektu sa má zostaviť, akú použiť *build* konfiguráciu a v akom prostredí sa má projekt spustiť. Pri vytvorení nového projektu sa pre každý cieľ automaticky vytvorí schéma, ktorá je pomenovaná podľa názvu projektu a obsahuje nastavenia na nasledujúce akcie:

- Spustenie aplikácie
- Spustenie testov vybraného cieľa
- Profilovanie výkonnosti aplikácie
- Vytvorenie analýzy zdrojového kódu
- Archivácia aplikácie na distribúciu pre testerov alebo na odoslanie do *App Store*

Každá akcia zostaví aplikáciu ako spustiteľný (*executable*) produkt. Na výber schémy a cieľa behu aplikácie slúži menu schémy (*Scheme menu*) v paneli nástrojov pracovného okna *Xcode*.

2.4.2 Výber cieľa behu aplikácie

Pri zostavovaní aplikácie, vybraný cieľ behu rozhodne, kde sa má aplikácia spustiť. Pre Mac aplikácie je tento cieľ rovnaké Mac zariadenie, na ktorom je aplikácia vytváraná. Pre iOS aplikácie je možnosť spustiť aplikáciu na iOS zariadení, ktoré je priradené do *Apple Developer Programu* a je pripojené k Macu, na ktorom je aplikácia vytváraná. Ďalšou možnosťou je iOS aplikáciu spustiť v simulátore, ktorý je nainštalovaný ako súčasť *Xcode* balíčka spolu s iOS SDK. Simulátor beží na Mac zariadení a simuluje prostredie iPhonu alebo iPadu.

Menu schémy (*Scheme menu*) 

ponúka na výber schému a cieľ behu aplikácie. Tieto dve nastavenia sú oddelené a teda schéma v sebe neobsahuje cieľ behu. Rovnaká schéma môže byť teda spustená na rôznych typoch simulátorov a iOS zariadení.

2.4.3 Beh aplikácie v simulátore

iOS simulátor dovoľuje simulovať niekoľko typov iPhonov a iPadov a na každom type môže bežať niekoľko typov iOS verzií. Ovládanie užívateľského rozhrania v simulátore prebieha pomocou myši a klávesnice, kde myš simuluje dotyky užívateľa. Pomocou klávesových skratiek je tiež možné dosiahnuť rotáciu zariadenia a iné akcie. V menu paneli iOS simulátora je možné použiť sekciu *Hardware* na:

- Rotáciu zariadenia v smera a proti smeru hodinových ručičiek
- Simulovanie zatrasenia zariadenia
- Zaslanie simulovanej udalosti upozornenia na málo pamäte (*low-memory warning*)
- Simulovanie geografickej polohy
- Simulovanie stlačenia *Home* alebo *Sleep* tlačidla

iOS Simulátor je základnou súčasťou vývojového procesu pred prechodom na testovanie na iOS zariadeniach. Dovoľuje otestovať správanie aplikácie ale jeho možnosti ako testovacej platformy sú oproti iOS zariadeniam obmedzené. V iOS simulátore nefungujú niektoré služby ako napríklad *Push notifikácie* alebo niektoré hardware funkcie ako *Camera*. Problematické je rovnako testovanie užívateľského rozhrania pre špecifické zariadenia, kedže iOS simulátor sa identifikue aplikácii ako *"x86"*, nie ako vybrané zariadenie.

2.4.4 Beh aplikácie na pripojenom zariadení

Na spustenie aplikácie na iOS zariadení ako *iPhone*, *iPod* alebo *iPad* je potrebné splniť niekoľko podmienok:

- Zariadenie musí byť pripojené k Mac počítaču
- Mac musí byť prihlásený do *Apple Developer Programu*
- Na Macu je nainštalovaný správny certifikát overovacej identity *Apple Developer Programu*
- iOS zariadenie je pridané na portáli *Apple Developer Programu*

Xcode dokáže vývojára previesť cez tieto vyžadované kroky takmer automaticky a zásah programátora potrebuje len málokedy. Všetky tieto kroky na spustenie aplikácie na iOS zariadení (ako *iPhone*, *iPad* alebo *iPod Touch*) sú potrebné vďaka extra bezpečnostnej vrstve implementovanej firmou Apple. Tá zabezpečí všetky užívateľské dáta a aplikáciu pred modifikáciou a následnou distribúciou treťou stranou. Počas vývoja developer vytvorí niekoľko prvkov, podľa ktorých je Apple následne schopný overiť identitu vývojára, jeho zariadení a produktov.

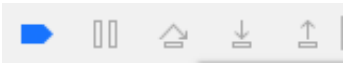
2.5 Ladenie aplikácie

Po spustení aplikácie tlačidlom *Run* z panela nástrojov pracovného okna *Xcode* sa po úspešnej kompilácii aplikácia spustí v *debug* móde, kde je možné ladenie priamo v editore zdrojového kódu. Oblasť ladenia (*Debug area*) v dolnej časti pracovného okna dovolí preskúmať stav aplikácie v danom čase, zobrazí hodnoty pre všetky premenné (*Quick Look*) a ponúka ovládacie prvky na kontrolu chodu ladenia aplikácie.

V oblasti navigátora sa počas ladenia zobrazí dopad aplikácie na systémové zdroje v reálnom čase. Sú tu zobrazené štatistiky využitia procesora, RAM pamäte, nnožstvo dát zapísaných a prečítaných z pevného disku zariadenia a dátový tok z a do aplikácie. Ak má aplikácia problémy s výkonom, jej ladenie sa dá spustiť v pomocnom programe *Xcode* s názvom *Instruments*, ktorý poskytuje podrobný pohľad na každý aspekt bežiacej aplikácie.

2.5.1 Breakpointy

Xcode dovoľuje prechádzať zdrojovým kódom riadok po riadku a zobrazovať tak stav aplikácie v akomkoľvek okamihu jej behu. V oblasti ladenia (*Debug area*) sú umiestnené ovládacie prvy na kontrolu chodu aplikácie. Sú tu rovnako zobrazené všetky momentálne premenné a výstup z konzoly. Vypnúť alebo zapnúť okno ladenia a konzolu je možné tlačidlami v pravom dolnom rohu oblasti ladenia ( ).



Obrázek 9: *Ovládacie prvky (Debug Controls)*

Pomocou *Play/Pause* tlačidla sa dá pozastaviť/opätovne spustiť vykonávanie aplikácie. Pre nastavenie breakpointu stačí kliknúť v editore zdrojového kódu vedľa čísla vybraného riadku kódu. Nastavený breakpoint reprezentuje modrá šípka.

Počas toho, ako je aplikácia pozastavená, aktuálne vykonávaný riadok kódu je zvýraznený zelenou farbou a je možné kontrolovať jej ďalší beh pomocou ovládacích prvkov *Step Over*, *Step Into*, *Step Out*.

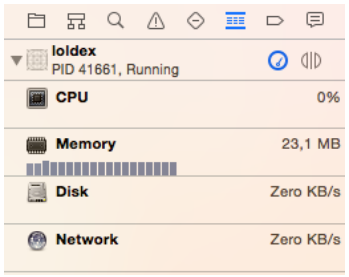
- *Step Over* vykonná aktuálne vyznačený riadok kódu, zahrňujúc všetky metódy ktoré tento riadok volá.
- *Step Into* začne vykonávanie vyznačeného riadku a pozastaví ho na prvom riadku volanej metódy.
- *Step Out* vykoná zvyšok prebiehajúcej metódy alebo funkcie.

2.5.2 Ladenie využívania systémových zdrojov

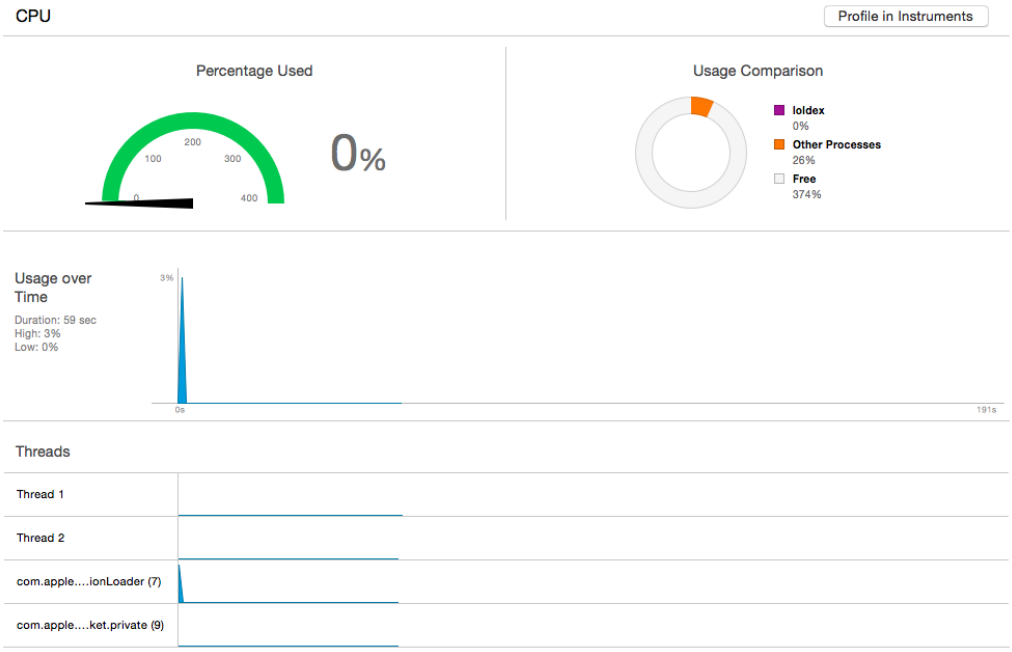
V oblasti navigátora sa počas behu aplikácie zobrazia hodnoty, ktoré poskytujú pohľad na jej výkonnosť. Napríklad po kliknutí na využitie procesora sa v sekcii editora zobrazí

podrobné grafické znázornenie jeho využitia. Tu je veľmi jedoduché určiť pri akej činnosti sa vyskytujú extrémne výkyvy vo využití systémových prostriedkov.

Pre dôkladnú hĺbkovú analýzu je možné použiť tlačidlo *Profile In Instruments* v oblasti editora výkonu. Táto akcia prepne bežiacu aplikáciu z*Xcode* do okna *Instruments* a pokračuje v behu, pričom začne ponúkať všetky podrobné dáta o svojom behu.



Obrázek 10: *Navigátor výkonnosti (Performance Navigator)*



Obrázek 11: *Detail zobrazenia výkonu*

2.5.3 Meranie výkonnu aplikácie

Instruments, aplikácia integrovaná do balíčka *Xcode*, zbiera dáta z bežiacej aplikácie a prezentuje ich v grafickej podobe časovej osi. *Instruments* dokážu zbierať informácie o výkone v rôznych oblastiach ako spotreba RAM pamäti, aktivita diskového úložiska, sieťová

aktivita a grafické operácie. Všetky tieto informácie môžu byť zobrazované spolu alebo v kombinácii vybraných oblastí. Takáto analýza pomôže identifikovať miesta, kde je možné zlepšenie aplikácie. *Instruments* tiež ponúkajú automatizované testovanie elementov užívateľského rozhrania iOS aplikácie.

Existuje niekoľko spôsobov, ako spustiť beh aplikácie v *Instruments*:

- Kliknutím na tlačidlo *Profile In Instruments* v oblasti editoru po kliknutí na *Debug Navigator*.
- Výberom Product » Profile z menu lišty *Xcode*
- Úpravou vybranej schémy a špecifikovaním *Instrumentu*

2.6 Testovanie aplikácie

Testovanie prebieha pomocou vytvorenia automatických testov, ktoré používajú funkcie aplikácie a testujú jej výkonnosť. Výsledky týchto testov a sledovaní sa zobrazujú v oblasti navigátora pod záložkou *Tests*.

Mac OS X Server ponúka tzv. *Xcode service*[8], ktorý automaticky spúšta vytvorené testy. Tieto testy sú vytvorené pomocou *Xcode* na vývojárskom počítači Mac a v podobe botov (*bots*) bežia na separátnom serveri. Okrem spúštania samotných testovacích jednotiek sa boti starajú o statickú analýzu zdrojového kódu, kompiláciu aplikácie, jej archiváciu a distribúciu pre testerov v *AppStore*. Tento mechanizmus zaisťuje, že aplikácia je stále zachovaná v stave vhodnom na vydanie v *release* verzii a v prípade vzniknutia chyby, boti dokážu kontaktovať osobu, ktorej kód danú chybu spôsobil.

2.6.1 Vytvorenie testov

Xcode podporuje dva typy testovania a to testovanie funkcionality a výkonu. Test funkcionality sa zameriava na funkčnosť kódu a výkonnostný test meria čas vykonania volaných metód. Každý test vytvorí tetsovacie prostredie, otestuje zamerané časti aplikácie a po ich dokončení vytvorené prostredie vymaže.

Najčastejšie používaným typom testov je testovanie základných jednotiek (*Unit Testing*). *Unit* v kóde predstavuje jeho najmenšiu testovateľnú časť. Obvykle to je jedna funkcia alebo metóda v danej triede alebo niekoľko metód, ktoré slúžia na vykonanie základného účelu aplikácie. Tento typ testu je používaný na zistenie úpadku výkonu aplikácie po implementácii nového kódu. V ideálnom prípade by sa mali tieto testy implementovať ako prvé a následne nové metódy, ktoré tieto testy úspešne prejdú. Pripojením výkonostného testu sa určuje ako dlho jednotlivé *unit* testy trvajú na rôznych typoch zariadení.

Vybraný test odskúša daný kus kódu v naprogramovanom smere alebo odmeria špecifikú časť výkonu aplikácie. Ak výsledok testu je rôzny od definovaného očakávaného výsledku, tento výsledok sa vráti ako chybný (*failed*). Spojenie niekoľkých testov sa nazýva *test suite*. Pri vytvorení nového projektu *Xcode* vytvorí *unit test* v schéme, ktorá sa

používa na kompiláciu aplikácie a rovnako vytvorí implementačný súbor s predefinovanými metódami *setUp, tearDown* a *testExample*. Pridaním požadovaných podmienok testu do týchto metód je možné viac špecifikovať beh testu a jeho očakávané výsledky.

Pre spustenie všetkých testov stačí vybrať z menu lišty *Xcode* Prouct » Test. V oblasti navigátora v sekcii *Test navigator* sa po spustení zobrazia všetky testy, ich stav a ich výsledok. Na pridanie ďalších testov stačí kliknúť na tlačidlo (+) ľavom dolnom rohu navigátora. Na zobrazenie zdrojového kódu vybraného testu stačí na vybraný test kliknúť a jeho zdrojový kód sa zobrazí v oblasti editora.

Ak aplikácia testom prejde úspešne, zobrazí sa zelená ikonka v tvare diamantu na pravej strane vedľa mena úspešného testu. Ak sa test nepodarí zobrazí sa vedľa mena červený diamant a chyba, pre ktorú sa test nepodaril bude zobrazená v sekcii navigátora pod záložkou *Issue navigator*. Pre zobrazenie len neúspešných testov je v ľavej dolnej časti navigátora pripravené filtrovacie tlačidlo s týmto úcelom. Po vyriešení chyby, pre ktorú bol test neúspešný, sa daný test dá spustiť po kliknutí na červenú ikonu diamantu.

3 Implementácia a vývojový proces aplikácie

Na demonštráciu vývojového procesu pre iOS platformu a jej možností slúži vlastná mobilná aplikácia pre hernú komunitu MOBA hry *League of Legends* s názvom *loldex - League of Legends Mobile Companion*. Primárnym cieľom aplikácie je poskytnúť neobmedzený prístup k aktuálnym herným dátam, stavu herných postáv a hráčov a prístup k najnovším udalostiam v hernej komunite.

3.1 Čo je League of Legends

League of Legends od vývojára Riot Games je momentálne jedna z najrozšírenejších *MOBA* hier na PC a Mac. MOBA (*Multiplayer Online Battle Arena*) je podžáner *real-time* strategických hier, kde hrajú proti sebe dva týmy s 5 hráčmi. Cieľom hry je zničiť budovu základne (*Nexus*) nepriateľského týmu. Každý hráč ovláda jednu postavu (*Champion*), kde každá z viac ako 120 postáv má svoje špecifické schopnosti a plní určitú úlohu v týme. V základoch *LoL* vychádza z najpopulárnejšej mapy *Defense of the Ancients* (DotA) pre RPG hru Warcraft III: The Frozen Throne od štúdia Blizzard Entertaiment.

LoL je hra s jednou s najväčších herných komunít na svete. Oficiálna Facebook [2] stránka má viac ako 12 miliónov členov, oficiálny Twitter účet[3] sleduje viac ako 2 milióny ľudí, na YouTube kanáli[4] má LoL 7 miliónov odberateľov a /r/leagueoflegends subreddit[5] na portáli Reddit má cez 650 tisíc členov. Na reddite je LoL komunita tou najväčšou z herných komunít, je vysoko aktívna a na príspevky často reagujú či vývojári Riot Games ako aj CEO Brandon "*Ryze*"Beck a President Marc "*Tryndamere*"Merrill.

Riot Games je tiež priekopníkom v rozširovaní verejného povedomia o *e-sports*. Vďaka tejto snahe USA od polovice roka 2013 uznáva profesionálnych LoL hráčov ako atlétov a udeľuje zahraničným hráčom pracovné víza s týmto titulom.[10]

Počas roka (sezóny) sa na každom kontinente každotýždenne odohrávajú regionálne majstrovstvá medzi profesionálnymi e-sports týmami o titul regionálneho šampióna. Na záver sezóny sa vždy na vybranom kontinente uskutoční svetový šampionát medzi najlepšími svetovými týmami. Všetky zápasy sú vysielané naživo online na portáloch Twitch.tv, Azubu.tv a rovnako aj na YouTube Live Events. Svetové majstrovstvá 2014, odohrávajúce sa počas 15 herných dní sledovalo spolu na týchto kanáloch viac ako 288 miliónov dívákov, v 19 jazykoch a najvyšší počet divákov v jeden čas bol viac ako 11 miliónov[9].

S často sa meniacimi menšími či väčšími časťami a tak aktívnou komunitou je potrebné, aby hráči mali k dispozícii nástroj, ktorý im umožní rýchly a ľahký spôsob prístupu k týmto zmenám, k novinkám z komunity a rovnako aj prehľad o hráčovom postupe v hre samotnej. *loldex* bol navrhnutý tak, aby tento prístup užívateľom a hráčom umožnil.

[2]Odkaz na stránku: https://www.facebook.com/leagueoflegends
[3]Odkaz na stránku: https://twitter.com/LeagueOfLegends
[4]Odkaz na stránku: https://www.youtube.com/user/RiotGamesInc
[5]Odkaz na stránku: http://www.reddit.com/r/leagueoflegends

Obrázek 12: Gameplay League of Legends

3.2 Návrh funkcií aplikácie

V nasledujúcom zozname sú uvedené všetky implementované funkcie:

1. **Champion list**
    - Aktuálny zoznam všetkých hrateľných postáv v hre, uložený lokálne na zariadení pre prístup k dátam v prípade nedostupnosti internetového pripojenia.
    - Zobrazený v tabuľke s menom postavy, jeho avatarom a pozadie bunky je časť jeho *splash artu*.
    - Obrazovka tohto listu je prístupná z úvodnej obrazovky aplikácie (*Root-View Controller*).
    - Po výbere určitej postavy sa zobrazí detail postavy.
2. **Champion detail**
    - Zobrazenie podrobných detailov o vybranej postave.
    - Všetky dáta sú uložené na zariadenie pre prípad potreby prístupu k dátam bez internetového pripojenia.
    - Ako pozadie obrazovky je použitý *splash art* vybranej postavy.
    - Pri scrollovaní pribudne na pozadí *blur* pre lepšiu čitateľnosť.
    - Zobrazené sú názov postavy, druhotné meno (*title*), cena v in-game mene (*IP - Influence points*) a cena v reprezentácii reálnej meny (*RP - Riot points*).

- V ďalšej časti sú zobrazené *ranked* štatistiky danej postavy, ktoré dajú užívateľovi pohľad na to, ktoré postavy sú najviac/najmenej hrané, ktoré postavy majú najvyššiu percentuálnu úspešnosť výhry (*win ratio*) a ktoré postavy sú percentuálne najviac blokované nepriateľským týmom (*ban ratio*).
- V ďalšej sekcii sú zobrazené schopnosti vybranej postavy s dôrazom na *theorycrafting* - v popisoch budú zobrazené čísla a percentuálne vyjadrenia, ktoré vyjadrujú rozsah škálovateľnosti danej schopnosti.

3. **Summoner profill**
   - Zobrazenie podrobných detailov o hráčovi.
   - Časť obrazovky je vyhradená pre zobrazenie štatistík ako Výhry, Prehry , *Win Ratio*, *Creepscore*, *K.D.A. Ratio* a to zvlášť pre hodnotené hry (*Ranked*) a klasické hry (*Normal*).
   - Ďalšia časť obrazovky je vyhradená na zobrazenie štatistík jednotlivých herných postáv (*Champions*) ako Výhry, Prehry, *Win Ratio*, *K.D.A. Ratio* a počet zaznamenaných *Doublekills*, *Triplekills*, *Quadrakills* a *Pentakills*.
   - Zobrazenie posledných 10 hier bez rozlišovania *normal/ranked* s možnosťou detailného popisu danej hry, s porovnaním medzi všetkými zúčastnenými hráčmi.

4. **Sales**

   League of Legends obsahuje *in-game* obchod, v ktorom je možné zakúpiť doplnky do hry pomocou jednou z dvoch herných mien. Influence Points (*IP*) hráč získava ako odmenu za hranie, Riot Points (*RP*) si hráč kupuje za skutočné peniaze. Každý utorok a piatok *Riot* uverejňuje 3 postavy a 3 zmeny vizuálu postavy (*Skin*), ktorých RP cena klesla o 50% do ďalšieho uverejňenia nových zľiav.
   - Zobrazenie aktuálnej zľavy.
   - Pozadie bunky je *splashart* skinu postavy.

5. **Patch notes**

   Vývojári *League of Legends* aktualizujú a upravujú samotnú hru v niekoľkotýždňových intervaloch. Každý *patch* prináša množstvo menších zmien schopností herných postáv ale aj obsiahle zmeny v mechanikách fungovania postáv a hry samotnej.
   - Zobrazenie kompletných zmien v aktuálnej verzii hry.
   - Zmeny sú rozdtriedené do sekcií jednotlivých herných postáv alebo podľa máp, pre ktoré sú dané zmeny určené.
   - Po výbere sekcie sa zobrazia detaily pre danú postavu/mapu.
   - Zdrojom dát bude vlastná REST implementácia z dôvodu nedostupnosti verejnej API zo strany *Riot Games*.
   - Tieto dáta sú ukladané a dostupné offline.

6. **Media Feed**
   Komunita *League of Legends* je jednou z najaktívnejších herných komunít vôbec. Media Feed funkcia by umožňovala užívateľovi sledovať tie najnovšie a najaktuálnejšie príspevky zo sociálnych sietí venovaných LoL.
   - Zobrazenie Top 25 príspevkov z subredditu */r/leagueoflegends*.
   - Najnovšie príspevky oficiálnych účtov League of Legends ako Facebook alebo Twitter.
   - Výber najakutálnejších Twitter príspevkov od najsledovanejších League of Legends účtov a pro hráčov.

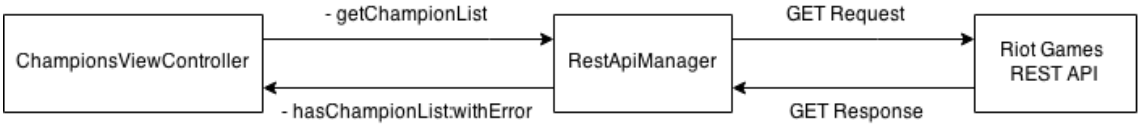
3.3 Implementácia funkcií

V nasledujúcej sekcii je zdokumentovaný postup implementácie definovaných funkcií v sekcii 3.2.

3.3.1 Champion list

Zoznam hrateľných postáv demonštruje implementáciu REST komunikácie s API *Riot Games* a prácu s iOS tabuľkou triedy *UITableView* a jej delegátom, použitie vlastnej bunky tabuľky *UITableViewCell*, prácu s *UIImage* a *UIImageView*, lazy load obrázkov zo vzdialenej lokácie(internet) a použitie notifikácií *NSNotificationCenter*.

Ako zdroj dát bude slúžiť objekt triedy NSDictionary (*NSDictionary* champions*) inicializovaný v triede AppDelegate a tým bude prístupný pre celú aplikáciu bez opätovnej potreby volať rovnakú REST metódu. V implementácii aplikácie sa vždy kontroluje stav tohto objektu pred začatím s operáciami nad týmto objektom a preto je aplikácia schopná zachovať sa korektne aj v prípade chybného objektu.



Obrázek 13: Bloková schéma toku dát pre ChampionsViewController

Tabuľka *tableView* je vložená do *ChampionsViewController*-a pomocou Interface Builderu, je tu nastavený jej *delegate* aj *dataSource* na objekt rodičovského controllera. Aby *tableView* korektne fungovala, musí sa stať *ChampionsViewController* delegátom aj zdrojom dát pre UITableView a je potrebné implementovať vyžadované delegát metódy pre UITableView.

- **- numberOfSectionsInTableView** nastaví počet sekcií tabuľky. V našom prípade je to 1.

- **- tableView:numberOfRowsInSection** nastaví počet buniek tabuľky. V našom prípade je to počet hrateľných postáv v na začiatku stiahnutom aktuálnom slovníku, teda *[[Utilities championList][@"ids"] count]*.

- **- tableView:cellForRowAtIndexPath:** nastaví vzhľad pre každú jednotlivú bunku tabuľky. Tu sa používa metóda *Reuse Identifier*, ktorá alokuje len toľko buniek, koľko je zobrazených v jednom čase v tabuľke a táto bunka sa uvoľní akonáhle opustí priestor obrazovky a objekt bunky je pripravený na opätovné použitie. Týmto spôsobom sa táto delegát metóda zavolá len toľkokrát, koľko buniek je zobrazených a volá sa dynamicky krátky čas predtým, ako sa zobrazí nová bunka a túto bunku nastaví.

Ako *view* objekt slúži pre každú bunku vlastná trieda *ChampionTableViewCell* s vlastným *.xib* súborom. V ňom sú vytvorené a umiestnené všetky objekty bunky tabuľky. Po vytvorení a inicializovaní bunky je potrebné pre každú nastaviť hodnoty UI objektov.

```
1  - (UITableViewCell *)tableView:(UITableView *)tableViewOfChampions cellForRowAtIndexPath:(
       NSIndexPath *)indexPath {
2      ChampionTableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"
           ChampionTableViewCell" forIndexPath:indexPath];
3
4      NSString* documentsDir = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
           NSUserDomainMask, YES) objectAtIndex:0];
5      cell.imageViewBackground.image = [UIImage imageWithContentsOfFile:[documentsDir
           stringByAppendingPathComponent:[NSString stringWithFormat:@"%@.jpg", [[Utility
           championList][@"ids"] objectAtIndex:indexPath.row]]]];
6      cell.championNameLabel.text = [NSString stringWithFormat:@"%@", [[Utility championList][@"
           names"] objectAtIndex:indexPath.row]];
7      cell.championTitleLabel.text = [NSString stringWithFormat:@"%@", [[Utility championList][@"
           titles"] objectAtIndex:indexPath.row]];
8      cell.backgroundColor = [UIColor clearColor];
9      return cell;
10 }
```

Výpis 1: Nastavenie dát pre bunku tabuľky

- **- tableView:heightForRowAtIndexPath:** nastaví výšku bunky tabuľky na danom indexe v bodoch (*points*).

- **- tableView:didSelectRowAtIndexPath:** sa zavolá po tom, čo užívateľ vyberie bunku tabuľky. Následne sa zavolá metóda, ktorá vykoná presun na ďalšiu obrazovku detailu vybranej hernej postavy. Tento presun je definovaný pomocou reťazca znakov *segueIdentifier* a je vytvorený v editore *Interface Builder*. Pri tomto presune sa volá metóda - *prepareForSegue:sender:* triedy *UIViewController*, ktorá je v triede ChampionsViewController preťažená, aby do detailu postavy predala identifikátor (*id*) vybranej postavy.

Efekt posúvania pozadia bunky je implementovaný za pomoci *UIScrollView* delegáta. Trieda *UITableView* z *UIScrollView* triedy dedí a to znamená, že scrollovací delegát funguje priamo aj pre objekt tabuľky.
Pomocou delegát metódy *-scrollViewDidScroll:* je zaistené, že pri každom pohybe (scrollovaní) tabuľky sa v triede vlastnej bunky vykoná kód na úpravu polohy obrázku pozadia bunky. Ako prostriedok na signalizáciu pohybu tabuľky pre vlastnú triedu bunky je použitá notifikačná správa v *NSNotificationCenter*. Druhou možnosťou signalizácie je využitie delegát metódy v *ChampionTableViewCell* a jej volanie pre každú viditeľnú bunku jednotlivo. Táto metóda bola však výrazne pomalšia. Preto bola na implementáciu tohto efektu použitá prvá metóda.

Prvým krokom v implementácii grafického efektu je potrebné zistiť smer pohybu tabuľky. Tento údaj sa určí pomocou porovnania predchádzajúceho uloženého offsetu obsahu tabuľky s tým aktuálnym. Po doterminovaní tohto údaju sa zavolá notifikácia v *NSNotificationCenter* s týmto údajom. Ako reakciu na túto notifikáciu vlastná trieda bunky tabuľky implementuje metódu, ktorá upraví *constraint* obrázku pozadia a tým ho posunie smerom pohybu tabuľky.

Obrázek 14: Champion List

3.3.2 Champion detail

Stránka detailu hernej postavy demonštruje používanie funkcie *Auto-Layout*, prácu s triedou typu *NSObject*, kreslenie objektov na canvas, používanie *Grand Central Dispatch* (GCD) a pridávanie rozpoznania dotykov (*Gesture recognizers*).

V detaile postavy je rodičovským objektom pohybujúci sa *UIScrollView*. Vďaka implementácii Auto-Layoutu je výška totho rodičovského *view* automaticky dopočítaná pomocou všetkých jeho objektov a ich *constraints*. Pri zmene výšky nejakého objektu sa výška UIScrollView zmení a užívateľ je schopný scrollovať až na koniec textu. Auto-Layout tiež zabezpečuje, že UI a všetky jeho objekty budú vždy zobrazené v natívnom rozlíšení a veľkostiach, čím je zabezpečená kompatibilita UI na zariadeniach s rôznymi uhlopriečkami.

Každý objekt v *ChampionDetailViewController* má 4 základné constrainty, ktoré určujú jeho polohu. *Leading* constraint určuje vzťah ľavej hrany vybraného objektu voči inému objektu. Vo väčšine prípadov je to hrana obrazovky alebo koniec iného objektu. *Top* constraint je vzťah medzi hornou hranou daného objektu a iným objektom a jeho zväčša spodnou hranou. *Trailing* constraint je vzťah medzi pravým okrajom vybraného objektu a začiatko iného objektu alebo hranou obrazovky. Štvrtým constraitom je *Height*, ktorý udáva výšku objektu v bodoch (*points*).

Na uchovanie všetkých informácií o vybranej postave slúži vytvorená trieda *Champion* typu *NSObject*. Inicializačná metóda tohto objektu vyžaduje slovník NSDictionary ako vstupný parameter. Následne pri inicializácii naplní jednotlivé *property* tohto objektu.

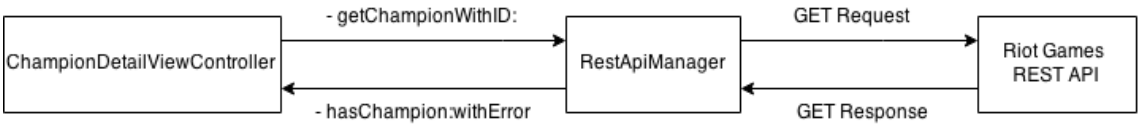
Na vykreslenie grafického efektu informačných *progress barov* je využité natívne iOS kreslenie rovných čiar a kriviek a ich následné renderovanie na canvas (*view*). Každá zobrazená čiara zobrazuje jednu z hodnôt uložených v objekte Champion. V objekte triedy *UIBezierPath* je uložený začiatočný a konečný bod čiary, ktorý sa nastaví ako cesta (*path*) pre objekt vrstvy *CAShapeLayer*. Tejto vrstve sa následne nastaví farba a jej šírka a pridá sa ako detská vrstva (*sublayer*) medzi vrstvy objektu *self.infoView*.

```
1  - (UIBezierPath *)getLineWithStartPoint:(CGPoint)start endPoint:(CGPoint)end {
2      UIBezierPath *line = [UIBezierPath bezierPath];
3      [line moveToPoint:start];
4      [line addLineToPoint:end];
5      return line;
6  }
```

Výpis 2: Implementácia vytvorenia objektu UIBezierPath

```
1  - (CAShapeLayer *)getLayerWithLine:(UIBezierPath *)line andStrokeColor:(UIColor *)strokeColor {
2      CAShapeLayer *lineLayer = [CAShapeLayer layer];
3      lineLayer.path = line.CGPath;
4      lineLayer.strokeColor = strokeColor.CGColor;
5      lineLayer.lineWidth = 18.0f;
6      return lineLayer;
7  }
```

Výpis 3: Implementácia vytvorenia vrstvy CAShapeLayer



Obrázek 15: Bloková schéma toku dát pre ChampionDetailViewController

Technológia GCD (Grand Central Dispatch)[12] bola vyvinutá firmou Apple na správu súbežných úloch na viacjadrových Mac OS a iOS zariadeniach. Tento prístup je výkonnejší ako použitie zámkov (*lock*) alebo vlákien (*threads*). Použitie asynchrónneho GCD v triede ChampionDetailsViewController bolo nutné z dôvodu výkonnostne náročnejšj úlohy sťahovania obrázkov z URL a ich následného zobrazovania do vytvorených *UIImageView* objektov. Vo fronte na pozadí (*dispatch_get_global_queue*) sa asynchrónne stiahne obrázok do lokálneho objektu typu NSData, ktorý sa potom pomocou metódy *imageWithData:* priradí do vybraného UIImageView. Z dôvodu že všetky úpravy UI musia byť volané z hlavnej fronty, toto priradenie zavoláme pomocou GCD z hlavného vlákna *dispatch_get_main_queue*.

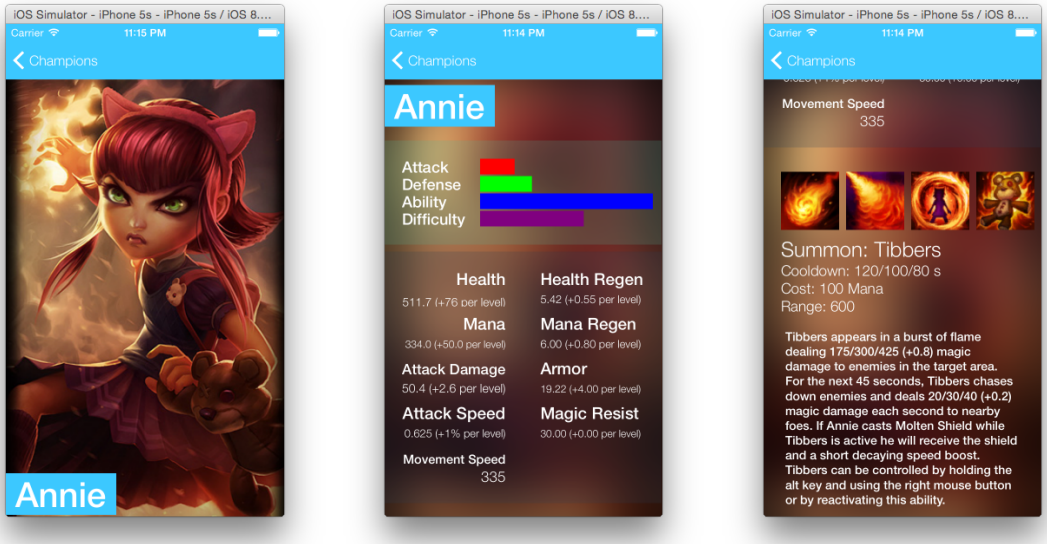
```
1  dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
2          for (int i = 0, i < 4, i++) {
3              imageData[i] = [NSData dataWithContentsOfURL:[NSURL URLWithString:[NSString
                   stringWithFormat:@"http://ddragon.leagueoflegends.com/cdn/%@/img/spell/%@.
                   png", [Utility getCurrentVersion], spellNames[i]]]];
4          }
5            dispatch_async(dispatch_get_main_queue(),^{
6                for (int i = 0, i < 4, i++) {
7                ((UIImageView*)spellImages[i]).image = [UIImage imageWithData:imageData[i]];
8                }
9            });
10 });
```

Výpis 4: Implementácia Grand Central Dispatch

Na prijímanie interakcií od užívateľa je pripravených niekoľko iOS Interface objektov ako sú tlačidlá, posuvníky alebo prepínače. V niektorých situáciách je však potrebné, aby dokázali reagovať na dotyk aj objekty, ktoré na to nie sú priamo vytvorené, ako napríklad objekt zobrazenia obrázka *UIImageView*. V tomto prípade je možné pridať na objekt rozpoznávanie dotyku resp. interakcie pomocou *Gesture Recognizer*. Ten je možné pridať priamo v Interface Builderi pomocou drag and drop gesta alebo pomocou zdrojového kódu v implementačnom súbore. V ChampionDetailsViewControlleri je recognizer vytvorený programovo.

```
1  – (void)addGestureToImageView:(UIImageView*)imageView {
2      UITapGestureRecognizer* singleTap = [[UITapGestureRecognizer alloc]initWithTarget:self
           action:@selector(spellImageTapped:)];
3      singleTap.numberOfTapsRequired = 1;
4      singleTap.delegate = self;
5      imageView.userInteractionEnabled = YES;
6      [imageView addGestureRecognizer:singleTap];
7  }
```

Výpis 5: Implementácia Tap Gesture Recognizer



Obrázek 16: Champion detail

3.3.3 Sales a Patch Notes

Pre implementáciu funkcii na zobrazenie aktuálnych zliav herných postáv a posledných zmien v hre je potrebná vlastná server-side aplikácia z dôvodu neposkytovania REST API pre tieto údaje zo strany Riot Games.

Server-side implementácia je realizovaná pomocou jazyka *Python* a je rozdelená na 2 časti. Prvá časť je REST API aplikácia, pomocou ktorej iOS aplikácia získava uložené dáta z MySQL databázy. Druhou časťou je *web scrapper* aplikácia, ktorá získava potrebné dáta z webových stránok *League of Legends*.

Server beží na Ubuntu 14.10. Oba python scipty sú spustené ako *service* pomocou deamona *upstart*. Tento spôsob zabezpečuje že scripty pobežia počas behu serveru, budú spustené pri každom sputení alebo reštarte serveru alebo pri neočakávanom ukončení daného scriptu. Konfigurácia týchto služieb je umiestnená v */etc/init*.

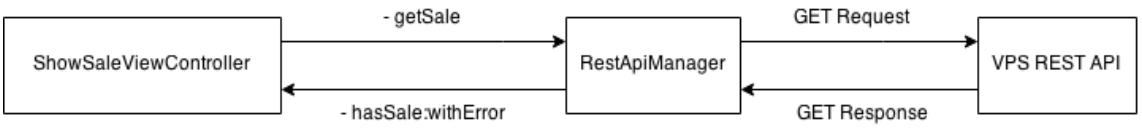
```
1 start on started networking
2 respawn
3
4 exec /home/loldex/rest.py
```

Výpis 6: Konfigurácia python scriptu ako *service*

```
1 $ sudo initctl reload-configuration
2 $ sudo start script
```

Výpis 7: Spustenie python scriptu ako *service*

REST server je implementovaný pomocou python knižnice *Flask*, beží na localhost adrese na porte 5000. GET metóda Sales na adrese */sales* vráti slovník v JSON formáte s potrebnými údajmi, ktoré sú uložené v MySQL databáze, s ktorou komunikuje vďaka knižnici *MySQLdb*.



Obrázek 17: Bloková schéma toku dát pre ShowSaleViewController

```
#! /usr/bin/python
__author__ = 'martinpucik'

from flask import Flask, url_for , jsonify
import MySQLdb
import sys
app = Flask(__name__)

@app.route('/sales', methods=['GET'])
def api_sales() :
    my_db = MySQLdb.connect(host="localhost", port=3306, user=USER, passwd=PW, db=DB)
    cursor = my_db.cursor()
    sql = "SELECT * FROM sales ORDER BY id DESC LIMIT 1"
    cursor.execute(sql)
    db_data = cursor.fetchone()
    my_db.close()

    data = {
        'skin1' :
        {'name': db_data[2], 'thumb': db_data[3], 'price' : db_data[4], 'sale_price' : db_data[5]},
        'skin2' :
        {'name': db_data[6], 'thumb': db_data[7], 'price' : db_data[8], 'sale_price' : db_data[9]},
        'skin3' :
        {'name': db_data[10], 'thumb': db_data[11], 'price' : db_data[12], 'sale_price' : db_data[13]},
        'sale_name': db_data[1]
    }

    resp = jsonify (data)
    resp.status_code = 200
    return resp

if __name__ == '__main__' :
    app.run(host='0.0.0.0' )
```

Výpis 8: Implementácia REST API v jazyku Python

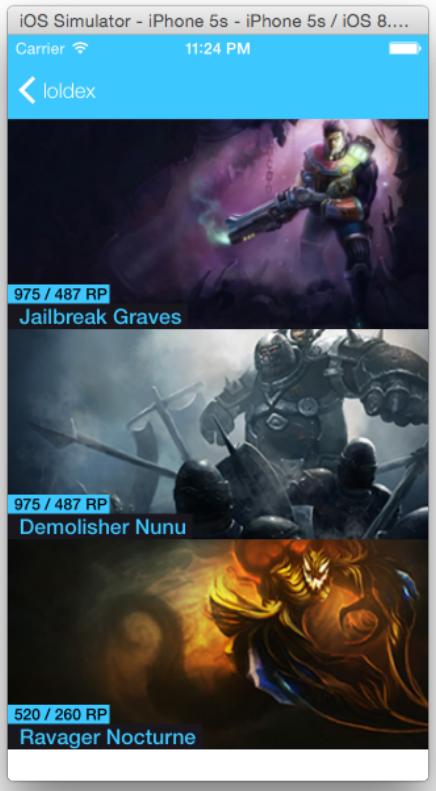
V iOS aplikácii sa o REST požiadavky stará trieda *RESTManager*, ktorá svojim delegátom posiela správy s obdržanými dátami v prípade úspešného volania REST metódy alebo chybovou hláškou.

```
1  - (void)getSale {
2      NSString* urlString = @"http://92.222.2.2:5000/sales";
3      NSURL* url = [NSURL URLWithString:urlString];
4      NSMutableURLRequest *req = [NSMutableURLRequest requestWithURL:url];
5
6      [NSURLConnection sendAsynchronousRequest:req queue:[NSOperationQueue mainQueue]
           completionHandler:^(NSURLResponse *response, NSData *data, NSError *
           connectionError) {
7
8          NSError* error = nil ;
9          NSDictionary* responseDict = [NSJSONSerialization JSONObjectWithData:data
10                                                                       options:kNilOptions
11                                                                         error:&error];
12         if (!error) {
13             if ([self.delegate respondsToSelector:@selector(hasSale:withError:)]) {
14                 [self.delegate hasSale:responseDict withError:nil];
15             }
16         } else {
17             if ([self.delegate respondsToSelector:@selector(hasSale:withError:)]) {
18                 [self.delegate hasSale:nil withError:error];
19             }
20         }
21     }];
22 }
```

Výpis 9: Implementácia volania REST metódy

Trieda *ShowSalesViewController* následne dáta rozparsuje do objektov triedy *SaleSkin* a uloží ich do poľa, z ktorého budú ľahko dostupné pri vytváraní *UITableView*.

Výška jednej bunky je nastavená na 160 bodov pomocou delegát metódy UITableView - *tableView:heightForRowAtIndexPath:*. Pomocou metódy - *tableView:cellForRowAtIndexPath:* je nastavená každá bunka tabuľky. Ako zdroj dát slúži objekt SaleSkin na rovnakom indexe v poli, na aký index bunky tabuľky bola táto metóda zavolaná. Pozadie bunky je asynchrónne stiahnuté pomocou *Grand Central Dispatch* z uloženej URL v objekte SaleSkin a je uložený v objekte typu UIImage, ktorý je následne v hlavnom vlákne zobrazený v objekte UIImageView užívateľského rozhrania.



Obrázek 18: Sales obrazovka

3.3.4 Media feed

Funkcia *Media Feed* implementuje zobrazenie príspevkov zo sociálnych sietí Reddit, Twitter a Facebook pomocou UITableView na úvodnej obrazovke aplikácie získaných z REST API, Facebook Graph iOS SDK a pomocou knižnice *STTwitter*. Tiež implementuje použitie vlastnej UITableViewCell bunky tabuľky na zobrazenie obsahu príspevku a funkciu iOS *snapshotViewAfterScreenUpdates*.

Pomocou funkcie *snapshotViewAfterScreenUpdates* je možné vytvoriť efekt bočného menu. *UIView* ktorý je v hierarchii objektov obrazovky najvyššie obsahuje tabuľku *UITableView* na zobrazenie príspevkov zo sociálnych sietí a tlačidlo *UIButton*, ktoré slúži na odsunutie tohto UIView na stranu a sprístupnenie bočného menu. V tomto UIView je rovnako pridaný rozpoznávač posuvných gest *UIPanGestureRecognizer*. Pri rozoznaní posunu zľava doprava v tomto UIView sa zavolá rovnaká metóda ako aj po stlačení tlačidla menu - *doSlideOut*, ktorá najskôr pomocou *snapshotViewAfterScreenUpdates* vytvorí nový subview aktuálneho stavu obrazovky a pridá ho ako najvyšší objekt UI a následne ho

posunie smerom k pravej hrane obrazovky o definovaný počet bodov *peakValue* a zobrazí tak bočné menu.

```
1  - (UIView *)getSnapShot {
2      return [[UIScreen mainScreen] snapshotViewAfterScreenUpdates:YES];
3  }
```

Výpis 10: Metóda na vytvorenie subview s aktuálnym stavom obrazovky

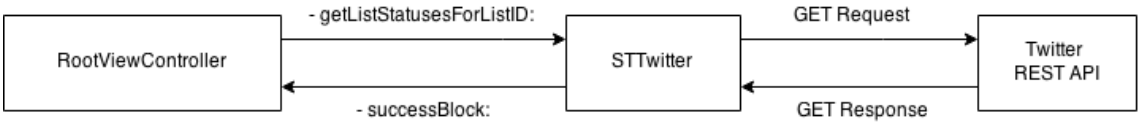
```
1  - (void)doSlideOut {
2      [self.snapshotView addSubview:[self getSnapShot]];
3      [self performSelector:@selector(setNeedsStatusBarAppearanceUpdate)];
4      [UIView animateWithDuration:0.5 animations:^{
5          [self.snapshotView setFrame:(CGRect){
6              peakValue, 0, self.snapshotView.frame.size
7       }];}];
8  }
```

Výpis 11: Efektu bočného menu

Bočné menu pozostáva z hernej ikony hráča a jeho IGN (In Game Name) a navigačných tlačidiel k jednotlivým funkciám aplikácie. Celá funkcionalita bočného Menu je realizovaná pomocou Interface Builderu a prechodov medzi obrazovkami typu *Push Segue*.

Implementácia komunikácie s Twitter API je riešená pomocou BSD-3 licencovanej knižnice *STTwitter*. Po jej pridaní do projektu táto knižica vyžaduje prítomnosť frameworkov *Accounts.framework*, *Social.framework*, *Twitter.framework*. Na autentifikáciu requestov je potrebná vytvorená aplikácia na portáli https://apps.twitter.com, kde po jej registrovaní sú vytvorené 2 kľúče, *Consumer Key* a *Consumer Secret*, ktoré slúžia ako autorizačná entita. Knižnica sa inicializuje pomocou metódy *twitterAPIAppOnlyWithConsumerKey:consumerSecret* a jednotlivé požiadavky na REST API sa volajú z *completion block*u metódy *verifyCredentialsWithSuccessBlock*.

Pomocou metódy - *getListsStatusesForListID:* sa po úspešnom requeste v bloku *successBlock* získa pole definovaného počtu posledných príspevkov vybraného listu účtov. Každý objekt v poli je typu JSON a je uložený v NSDictionary. Z týchto slovníkov aplikácia následne vyparsuje potrebné informácie ako text príspevku, prípadné obsiahnuté URL odkazy, meno účtu, profilový obrázok a dátum vytvorenia príspevku a uloží ich do objektu vlastnej triedy *Tweet*. Tieto objeky zoskupí v poli *tweets*, ktoré bude slúžiť ako zdroj dát pre delegát metódy UITableView, v ktorej budú príspevky zobrazené.



Obrázek 19: Bloková schéma toku dát pre twitter knižnicu STTwitter

```
1  [ twitter  verifyCredentialsWithSuccessBlock:^(NSString *bearerToken) {
2          [ twitter  getListsStatusesForListID:TW_LIST_ID
3                                       sinceID: nil
4                                       maxID:nil
5                                       count:NUMBER_OF_TWEETS
6                                       includeEntities :NO
7                                       includeRetweets:NO
8                                  successBlock:^(NSArray* statuses){
9                                  for  (NSDictionary* status in statuses)  {
10                                        [self  parseTwitterStatus:status ];
11                                 }
12         } errorBlock :^( NSError* error){
13          [self  performSelector:@selector(loadTwitter) withObject:self afterDelay:10];
14     }}];
15      } errorBlock :^( NSError *error)  {
16           [self  performSelector:@selector(loadTwitter) withObject:self afterDelay:10];
17      }];
18 }
```

Výpis 12: Získanie príspevkov z Twitter listu

Na získanie príspevkov z Facebook stránky je nutné vytvorenie aplikácie na adrese *https://developer.facebook.com/apps*. Po vytvorení aplikácie je vytvorený identifikátor aplikácie *App ID* a privátny kľúč *App Secret*, pomocou ktorých sa pri spustení aplikácie vygeneruje *Access Token*, ktorým sa autorizujú volania Facebook Graph API. Z bezpečnostných dôvodov sa nemôže žiadať tento token priamo z aplikácie. Pre svoje fungovanie GET metóda Graph API na získanie tokenu potrebuje App ID aj App Secret vo svojom requeste. Ak by boli tieto dáta uložené priamo v aplikácii, boli by tak vystavené hrozbe zneužitia tretích strán. Na získanie Access Tokenu sa teda používa server-side Python script, ktorý ho získa a REST metóda */fbat* ho vráti ako *hash string* ktorý bol vytvorený z tokenu pomocou algoritmu sha256. Aplikácia následne použije rovnaký typ algoritmu na získanie tokenu, ktorý uloží ako premennú pre triedu RESTManager, na ďalšie použitie pri volaní Facebook Graph API.

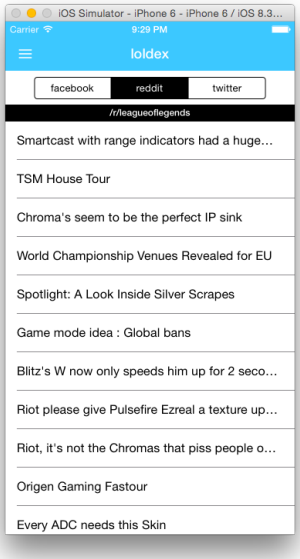
Pre získanie príspevkov z Facebook profilu *League of Legends* je v RESTManager triede implementovaná metóda - *getFacebookPosts*, ktorá pošle GET request s Graph API cestou */page-id/posts*. Po jeho úspešnom vykonaní je odpoveďou slovník *NSDictionary*, ktorý obsahuje dáta pre posledných 25 príspevkov, ak v requeste nebolo špecifikované inak.

Rovnako ako v prípade Twitter príspevkov, sa jednotlivé záznamy slovníka uložia do objektu typu *FacebookPost*, ktorý sa pridá do poľa, ktoré bude slúžiť ako zdroj dát pre UITableView tabuľku, ktorá príspevky zobrazí.

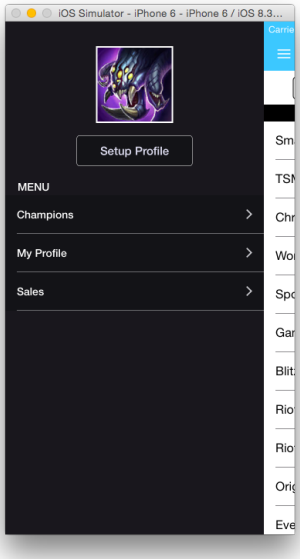
Ako ovládací prvok na zmenu zdroja príspevkov bol použitý objekt UISegmentControl vytvorený pomocou *Interface Builder* editora, s prepojením eventu typu *Value Changed* do metódy - *(IBAction)didChangeSource:(id)sender*. Táto metóda sa zavolá vždy, keď užívateľ vyberie jednu z troch možností zdrojov. Po zavolaní sa pomocou GCD metódy *dispatch_async* asynchrónne aktualizujú príspevky vybraného zdroja a po dokončení aktualizácie sa znovu načíta obsah UITableView tabuľky pomocou metódy - *reloadTable*.

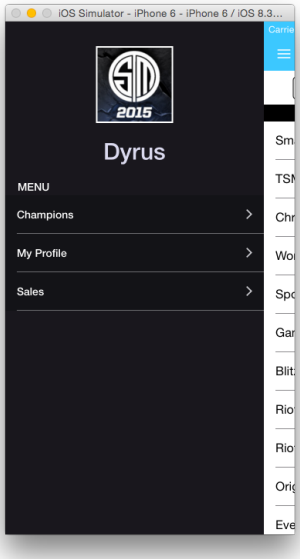
```
1  - (void)getFacebookPosts {
2      NSString* urlString = [NSString stringWithFormat:@"%@/leagueoflegends/posts?access_token
           =%@", FB_GRAPH_ADDRESS, fbAccessToken];
3      NSString *encodedURLString = [urlString stringByAddingPercentEscapesUsingEncoding:
           NSASCIIStringEncoding];
4
5      NSURL* url = [NSURL URLWithString:encodedURLString];
6      NSMutableURLRequest *req = [NSMutableURLRequest requestWithURL:url];
7
8      [NSURLConnection sendAsynchronousRequest:req queue:[NSOperationQueue mainQueue]
           completionHandler:^(NSURLResponse *response, NSData *data, NSError *
           connectionError) {
9
10         NSError* error = nil ;
11         NSDictionary* responseDict = [NSJSONSerialization JSONObjectWithData:data
12                                                                      options:kNilOptions
13                                                                        error:&error ];
14         if (responseDict) {
15             NSArray* posts = responseDict[@"data"];
16             for (NSDictionary* post in posts) {
17             [self parseFacebookPost:post];
18         }
19         } else
20          [self performSelector:@selector(loadFacebook) withObject:self afterDelay:10];
21     }];
22 }
```

Výpis 13: Získanie príspevkov League of Legends Facebook stránky



Obrázek 20: Media Feed





Obrázek 21: Bočné menu

3.4 Distribúcia aplikácie

Ako posledný krok vo vývojovom procese aplikácie je distribúcia aplikácie do AppStore obchodu. Pred samotným nahraním binárnej reprezentácie aplikácie je potrebné vykonať niekoľko krokov.

3.4.1 Apple Developer Program

Pre distribúciu iOS aplikácie je potrebné mať aktívny iOS Apple Developer Program. S aktívnym vývojárskym účtom je možný prístup k zdrojom a portálom, ktoré sú potrebné na úkony potrebné k distribúcii. S týmto vývojárskym účtom je nutné prihlásenie v nastaveniach Xcode v sekcii *Accounts*. Po úspešnom prihlásení je Xcode schopný zautomatizovať veľké množstvo úloh, ako napríklad podpisovanie aplikácie, používanie správneho *Provisioning Profile*, Development alebo Production certifikátu alebo správneho *App ID*.

3.4.2 Nastavenie projektu

Konfigurácia projektu musí obsahovať niekoľko povinných nastavení ako napríklad správny *bundle indetifier*, ktorý slúži na jednoznačné idetifikovanie aplikácie. Bundle ID musí byť UTI (*uniform type identifier*) reťazec znakov obsahujúci alfanumerické znaky (A-Z, a-z, 0-9), pomlčku (-), a bodku (.) v *reverse-DNS* formáte. Pre aplikáciu loldex je Bundle ID

sk.martinpucik.loldex. Toto ID sa musí zhodovať s položkou Bundle Identifier na vývojárskych portáloch *Members Center* a *iTunes Connect*.

Ďalšou povinnou položkou je vyplnený korektný vývojársky tým do ktorého je vývojár prihláseny v nastaveniach Xcode a pomocou ktorého sa bude aplikácia publikovať.

Sekcia *Deployment Info* špecifikuje minimálnu verziu iOS, ktorá je potrebná k behu aplikácie, na akých typoch zariadení a v akých orientáciách bude aplikácia bežať a ktorý *.storyboard* súbor je načítaný pri spustení aplikácie. Podľa typov vybraných zariadení môže byť aplikácia iPhone App, iPad App alebo Universal App.

App Icons and Launch Images sekcia sa používa na vloženie ikon a obrázkov pri spustení aplikácie. Sekcia je riešená špeciálnym typom editoru tzv. *Assets Catalog*, v ktorom je možné jednoducho pomocou drag and drop gesta vložiť požadované obrázky. Ikony majú pre rôzne typy zariadení, displejov a miesta zobrazenia špecifikované rozlíšenie uvedené v bodoch. Pre non-retina displeje (postfix 1x) sa počet bodov rovná počtu pixelov. Pre retina displeje sa počet pixelov zdvojnásobí oproti počtu definovancýh bodov (postfix 2x). Jediné zariadenie s trojnásobným počtom pixelov je iPhone 6Plus (postfix 3x). Pri vložení nesprávneho obrázku ikony alebo obrázku pri spustení to Xcode oznámi varovnou hláškou (*Warning*). Bez ošetrenia týchto varovaní nebude možné aplikáciu úspešne distribuovať.

3.4.3 Capabilities

Sekcia *Capabilities*[6] v nastaveniach cieľa projektu zobrazuje prehľad všetkých Apple technológií, ktoré sú v projekte obsiahnuté. Každá technológia sa dá jednoducho pridať alebo odobrať pomocou On/Off prepínača. Pri vypnutej technológii sa zobrazujú požiadavky, ktoré musia byť splnené aby daná technológia korektne fungovala. Pri zapnutej technológii sa zobrazuje jej aktuálny stav v projekte a prípadné chyby alebo nesplnené požiadavky pre správne fungovanie danej technológie. Požiadavkom na úspešnú distribúciu je vyriešenie všetkých uvedených chýb v tejto sekcii.

3.4.4 iTunes Connect

Ďalším krokom v distribučnom procese je vytvorenie záznamu aplikácie vo vývojárskom portáli *iTunes Connect*.[7] Po príhlásení[8] pomocou vývojárskeho účtu s aktívnym Apple Developer Programom sa sprístupní úvodné okno portálu[9].

Pred nahratím binárneho súboru aplikácie je potrebné mať v sekcii *Manage Your Apps* vytvorený záznam distribuovanej aplikácie. Tento záznam je vytvorený pomocou *Add New App*[10]. Prvá časť vytvorenia záznamu novej aplikácie vyžaduje vyplnenie mena aplikácie, hlavného (*default*) jazyka aplikácie, SKU identifikátora a výber Bundle ID vytvoreného

[6] Obrázok v prílohe: Capabilities
[7] Odkaz na stránku: https://itunesconnect.apple.com
[8] Obrázok v prílohe: SubmitScreenshot01
[9] Obrázok v prílohe: SubmitScreenshot02
[10] Obrázok v prílohe: SubmitScreenshot03

na portáli *Members Center*. Toto Bundle ID musí byť zhodné s Bundle ID vybranom v nastaveniach projektu aplikácie.

Pokračovaním na ďalšiu obrazovku[11] je vyžadované nastavenie dátumu, od ktorého bude aplikácia dostupná k predaju. *Price Tier* vyjadruje pomocou prevodných tabuliek za akú cenu sa bude aplikácia predávať v obchodoch jednotlivých krajín a je dovoľuje nastaviť automatickú zmenu ceny na vybrané časové obdobia.

Nasledujúcim krokom v príprave záznamu aplikácie je obrazovka[12], kde je potrebné vyplniť verziu aplikácie v rovnakej hodnote, aká je uvedená v nastaveniach projektu aplikácie. Je tu možné vybrať 2 kategórie, pod ktoré bude aplikácia spadať, pričom povinnou je len prvá kategória. Sekcia *Rating* vyžaduje pre každú jej podkategóriu určiť, v akej miere sa v aplikácii vyskytujú dané prvky. Na základe ich výskytu potom portál určí minimálny doporučený vek pre používanie aplikácie.

Sekcia *Metadata*[13] vyžaduje vloženie popisu aplikácie s vymenovanými funkciami. *Keywords* vyžaduje kľúčové slová, podľa ktorých bude vyhľadávací engine AppStore aplikáciu indexovať a zaraďovať do výsledkov hľadania. Maximálny počet znakov je 100 a jednotlivé kľúčové slová alebo frázy musia byť oddelené čiarkou. *Support URL* je povinná položka, ktorá obsahuje webovú stránku podpory aplikácie. Ostávajúce položky *Marketing URL* a *Privacy Policy URL* sú voliteľné.

Po uložení týchto informácií sa vytvorí záznam aplikácie a nastaví sa jej stav na *Prepare for Upload*. V detaile aplikácie je nutné pripraviť aplikáciu na nahranie binárnej podoby súhlasením s exportom. Tento proces sa spustí pomocou tlačidla *Ready to Upload Binary*.

Export Compliance[14] sa skladá z potvrdenia, že aplikácia používa alebo nepoužíva šifrované dáta. V prípade loldexu žiadne šifrované dáta nie sú. V inom prípade, keď bolo použité vstavané šifrovanie *CommonCrypto* sa môže vybrať možnosť *Yes* bez ďalších možností. V prípade že bolo použité iné šifrovanie, ako napríklad SRTP alebo ZRTP, je nutné mať potvrdený formulár od Amerického dozorného úradu.

Sekcia *Content Rights* vyžaduje potvrdenie, či bol v aplikácii použitý obsah tretích strán. Ak áno, je nutné potvrdiť že tento obsah je dostupný v tých teritóriách, v ktorýh bude prístupná aj aplikácia. Vývojárske štúdio League of Legends, Riot Games dovoľuje použitie ich Intellectual Property, pokiaľ bude projekt spĺňať ich podmienky používania. Hlavnými bodmi sú podmienky, že projekt má byť zdarma a verejné dostupný pre komunitu. Po úspešnom uložení tejto sekcie sa zmení stav aplikácie na *Waiting For Upload*. V tejto chvíli je portál iTunesConnect pripravený na nahranie binárneho súbory aplikácie.

3.4.5 Distribúcia z Xcode

Po tom ako je stav aplikácie na portáli iTunesConnect *Waiting For Upload*, je možné nahrať binárny súbor aplikácie na portál pomocou vstavanej funkcie *Application Loader*. Ako prvý krok je nutné odpojiť od Mac zariadenia všetky mobilné iOS zariadenia a následne vo výbere zariadení na ktorom sa spustí aplikácia zvoliť základnú položku *iOS Device*.

[11] Obrázok v prílohe: SubmitScreenshot04
[12] Obrázok v prílohe: SubmitScreenshot05
[13] Obrázok v prílohe: SubmitScreenshot06
[14] Obrázok v prílohe: SubmitScreenshot07

Týmto krokom sa v lište Menu sprístupní položka *Archive*[15]. Touto položkou Xcode spustí vytvorenie archívu resp. binárnej podoby aplikácie.

Po dokončení tvorby archívu sa automaticky otvorí okno *Organizer*[16] aplikácie Xcode, ktoré slúži okrem iného aj na prehľad a správu vytvorených archívov. Tu sa zobrazí práve vytvorený archív aplikácie *loldex*. Možnosťou *Distribute...* sa spustí príprava na nahranie archívu na portál iTunesConnect.

Prvá obrazovka nahrávacieho procesu definuje spôsob distribúcie aplikácie[17]. Prvou možnosťou, ktorou je distribuovaná aj aplikácia loldex, je distribúcia do iOS App Store. Ďalšou možnosťou je distribúcia vytvorenej aplikácie len vrámci firmy, kde bude aplikácia podpísaná vývojárom ale nebude nahraná do iOS App Store. Jej distribúciu do zariadení zabezpečuje daná firma. Treťou možnosťou je vyexportovať archív aplikácie do binárneho súboru.

Nasledujúcim krokom je výber *Provisioning Profile*-u, pomocou ktorého sa overí identita vývojára a zariadenia Mac z ktorého sa nahrávanie uskutočňuje[18].

Po výbere možnosti *Submit* sa začne samotný nahrávací proces[19], kde Application Loader overí všetky potrebné informácie a ich správnosť naprieč vývojárskymi portálmi. Ak tento proces prebehne bez chýb, skončí stavom *Submission Succeeded*[20]. V tomto momente sa stav záznamu aplikácie na portáli iTunesConnect zmení na *Upload Received*[21].

Po dokonční nahrávania je potrebné určitý čas počkať na finalizáciu spracovania daného binárneho súboru na portáli iTunes Connect. Keď bude daný súbor pripravený, stav aplikácie sa zmení *Waiting For Review*. Tento stav znamená, že aplikácia je zaradená do poradovníka a čaká na schválenie zo strany Apple Review týmu.

[15] Obrázok v prílohe: SubmitScreenshot08
[16] Obrázok v prílohe: SubmitScreenshot09
[17] Obrázok v prílohe: SubmitScreenshot10
[18] Obrázok v prílohe: SubmitScreenshot11
[19] Obrázok v prílohe: SubmitScreenshot12
[20] Obrázok v prílohe: SubmitScreenshot13
[21] Obrázok v prílohe: SubmitScreenshot14

4 Zhodnotenie

4.1 Zhodnotenie vývoja

Vývoj aplikácie bol zahájený v septembri 2013 a finalizovaný 21. 5. 2014 schválením aplikácie pre predaj v App Store.

Prvý mesiac vývoja bol sústredený na definíciu funkcií aplikácie, návrh užívateľského rozhrania, výber backend technológií a kolekciu potrebných informácií ohľadom dostupnosti herných dát ako aj legálne podmienky používania týchto zdrojov.

Implementácia definovaných funkcií bola dokončená po 4 mesiacoch kedy bola aplikácia presunutá z *alpha* to *beta* verzie a backend časť aplikácie bola nasadená na VPS.

Počas trvania beta fázy bolo implementované finálne užívateľské rozhranie a prevedené testy stability aplikácie a testy dostupnosti a spoľahlivosti jej *backend* časti. Po dosiahnutí požadovaných výsledkov bola aplikácia už v *release* verzii 1.0 dňa 18. 5. 2013 oficiálne odovzdaná na schvaľovací proces do App Store a po 2 dňoch bola schválená a uvedená do stavu *Ready for Sale*[22].

Za prvé 2 týždne dostupnosti dosiahla aplikácia viac ako 400 nákupov[23] bez akéhokoľvek prispenia marketingu. Aplikácia nezaznamenala žiadne kritické chyby a mala vysoké hodnotenia a pozitívny *feedback* od užívateľov[24].

Pre drobné zmeny v REST API zo strany *Riot Games* bolo nutné vydať 3 aktualizácie (verzia 1.0.1, 1.0.2 a 1.0.3), ktoré tieto zmeny implementovali, pričom tieto aktualizácie neznížili stabilitu aplikácie.

4.2 Zhodnotenie platformy

iOS je jedným s najrozšírenejších a najkomplexnejších mobilných operačných systémov súčastnosti. Jeho najväčšími kvalitami je jeho rýchlosť, spoľahlivosť, vysoká výkonnosť, nenáročné ovládanie, detailne prepracovaný dizajn užívateľského rozhrania a z neho plynúca vysoká úroveň *User Experience*.

Fakt že software aj hardware iOS zariadení je vytváraný jedine firmou Apple zabezpečuje maximálnu výkonnosť a kompatibilitu týchto zariadení s iOS systémom, minimálnu fragmentáciu medzi rôznymi veľkosťami displejov alebo obsahovanými funkciami a možnosťami týchto zariadení. Rovnako dovoľuje vývojárom menej sa sústrediť na optimalizáciu a kompatibilitu aplikácie na rôzne zariadenia a venovať viac času implementácii samotnej aplikácie.

Uzatvorenosť systému a *sandbox* umožňuje nielen užívateľom ale aj vývojárom dosiahnuť maximálne zabezpečenie svojich dát a zachovať pri tom vysokú možnosť rozšíriteľnosti a prispôsobenia.

Apple vývojárom v Apple Developer Programe ponúka prvotriedne vývojové nástroje, špičkový hardware, kompletnú, podrobnú a rozsiahlu dokumentáciu celého dostupného SDK s mnohými ukážkovými príkladmi implementácie, detailné návody a špecifikácie

[22] Obrázok v prílohe: Status história
[23] Obrázok v prílohe: Štatistika predajov
[24] Obrázok v prílohe: Reviews

pravidiel a doporučených postupov a implementácií. Početná a veľmi aktívna komunita vývojárov prispieva k rýchlym opravám chýb ako v iOS tak aj v jeho SDK a prostredníctvom diskusného fóra Apple Developer Programu ponúka rýchlu pomoc pri riešení problémov z implementáciou alebo funkčnosťou softwaru.

V neposlednom rade je z vývojárskeho hľadiska veľmi dôležitá schopnosť platformy generovať príjmy (*revenue*) a alternatívne možnosti monetizácie aplikácie. Vzhľadom na uzatvorený systém je možnosť nainštalovania pirátskej kópie aplikácie bez oficiálneho nákupu veľmi obmedzená a preto je iOS platforma oveľa efektívnejšia v produkovaní zisku ako konkurenčné platformy. Pri *freemium* modeli aplikácie, kde je aplikácia zdarma a jej funkcie sú spoplatnené pomocou In-App nákupov, je možnosť pirátstva nulová.

Pre svoju komplexnosť, pokrokovosť, detailné spracovanie, množstvo kvalitných funkcií a veľký potenciál je iOS volený ako primárna platforma pre väčšinu nielen jednotlivých mobilných vývojárov, *indie* vývojárskych štúdií ale aj pre veľké firmy s technickým ako aj bez technického zamerania a svetové herné vydavateľstvá a svojou prácou tak napomáhajú k vytvoreniu toho najpokročilejšieho mobilného operačného systému na svete.

Martin Púčik

5 Reference

[1] Lipsman, Andrew. *Apps Now Drive Half of All Time Spent on Digital* [online]. 25. 6. 2014.
<http://goo.gl/Z9Yp3e>

[2] International Data Corporation. *Press Release prUS25450615* [online]. 24. 2. 2014.
<http://goo.gl/CVRwYZ>

[3] App Annie. *App Annie Index: 2014 Retrospective* [online] 28. 1. 2015.
<http://goo.gl/5SyeZN>

[4] Apple Inc. *Program Enrollment - iOS Developer Program - Support - Apple Developer* [online]. c2015.
<https://developer.apple.com/support/ios/enrollment.php>

[5] Apple Inc. *App Distribution Quick Start* [online]. 20. 10. 2014.
<http://goo.gl/G38w8W>

[6] Apple Inc. *App Distribution Guide* [online]. 8. 4. 2015.
<http://goo.gl/njZoWr>

[7] Apple Inc. *iTunes Connect Developer Guide* [online]. 9. 4. 2015.
<http://goo.gl/2Rk2jX>

[8] Apple Inc. *What is Xcode service? - Server Help* [online]. c2015.
<http://goo.gl/QJGUWy>

[9] Gafford, Travis. *League of Legends 2014 World Championship Viewer Numbers (Infograph)* [online]. 1. 12. 2014.
<http://goo.gl/xOVcis>

[10] Eddie Makuch. *US government recognizes League of Legends players as pro athletes* [online]. 12. 07. 2013.
<http://goo.gl/TCkXt2>

[11] Riot Games. *Legal Jibber Jabber — Riot Games* [online]. 24. 11. 2012.
<http://www.riotgames.com/legal-jibber-jabber>

[12] Apple Inc. *Grand Central Dispatch (GCD) Reference* [online]. 17. 9. 2014.
<http://goo.gl/LaOz0g>

[13] Conway, Joe. *iOS Programming: The Big Nerd Ranch Guide.* 1. vyd. Big Nerd Ranch Guides. 2013. ISBN: 978- 0321942050.